

**IITK NETWORK -
HARDWARE AND SOFTWARE DEVELOPMENT ON IBM-1800**

**A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY**

**BY
SANJIV KUMAR**

**to the
COMPUTER SCIENCE PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
JULY, 1980**

I.I.T. KANPUR
CENTRAL LIBRARY

Vol. No. A 63037


11 AUG 1980

CSP-1580-M-KUM-HAR

CERTIFICATE

This is to certify that the thesis entitled, 'IITK NETWORK -
HARDWARE AND SOFTWARE DEVELOPMENT ON IBM-1800', which is being
submitted by Sri Sanjiv Kumar in partial fulfilment of the re-
quirements for the award of the degree of MASTER OF TECHNOLOGY,
has been carried out under my supervision and has not been sub-
mitted elsewhere for the award of a degree.

Kanpur
July 1980


A.S. Sethi
Assistant Professor
Computer Science Program
Indian Institute of Technology
Kanpur

ACKNOWLEDGEMENT

I express my deep gratitude to Dr. A.S. Sethi for suggesting me this project and guiding excellently throughout the work.

Thanks are due to Messrs Waryam Singh, V. Gupta, Dureja, S.L. Agarwal and S.A. Siddiqui, who were always ready to help me. I am grateful to the faculty and students of the Computer Science Programme for their cooperation and help throughout this work.

I thank Mr. H.K. Nathani for his excellent typing, Mr. Maikoo Lal for xeroxing and Mr. Shankar Bux Singh for the cyclostyling work.

Finally, I express my sincere thanks to Mr. R.K. Jain for his great cooperation and help during the work.

Kanpur
July 1980

- Sanjiv Kumar

ABSTRACT

A project was started at the Computer Centre, IIT Kanpur in 1978 to link up DEC-1090, TDC-316 and IBM-1800 in a star configuration using a MICRO-78 as the central switch. In the first phase of the work, hardware interfaces were designed for these machines (except the interfaces for the MICRO to DEC link). In the second phase, i.e., the current one, the hardware interfaces for the MICRO to DEC link have been designed, certain changes have been made in the earlier interfaces and a part of the software has been implemented on MICRO-78 and IBM-1800. This thesis describes the changes made in the earlier hardware interfaces and the software implementation on IBM-1800.

CONTENTS

Chapter 1	INTRODUCTION	1
Chapter 2	MODIFICATIONS IN EARLIER HARDWARE INTERFACES	<u>6</u>
Chapter 3	DESCRIPTION OF DDCMP	12
Chapter 4	IMPLEMENTATION OF DDCMP ON IBM-1800	29
Chapter 5	CONCLUSION	56
Appendix I		58

CHAPTER 1

INTRODUCTION

When two or more computers are interconnected to communicate among themselves, such a configuration is called a computer network. Computer networks have got a variety of applications, some of the most common ones being resource sharing, remote job entry, load sharing, etc. In India, not many networks have yet come up, but this field has started catching the interest of computer professionals in this country. IIT Kanpur was among the first to start developing an experimental computer network and this thesis describes a part of the work done on the development of this network. The following section gives a brief description of the IIT Kanpur network.

1-1. IIT/KANPUR COMPUTER NETWORK - A Brief Description

The proposed network for IIT/Kanpur will have DEC-1090, IBM-1800 and TDC-316 interconnected in a star fashion with a MICRO-78 as the central switch as shown in Figure 1-1. Following are some of the proposed applications of this network:

1. TDC-316 can be used as a remote job entry station for DEC-1090.
2. Card punch on IBM-1800 can be used by users of DEC-1090 and TDC-316.
3. 7-track to 9-track conversion of the data on magnetic tapes can be done since the 1800 has got a 7-track tape drive and the DEC has got a 9-track drive.
4. TDC-316 can be used as a concentrator for terminals.

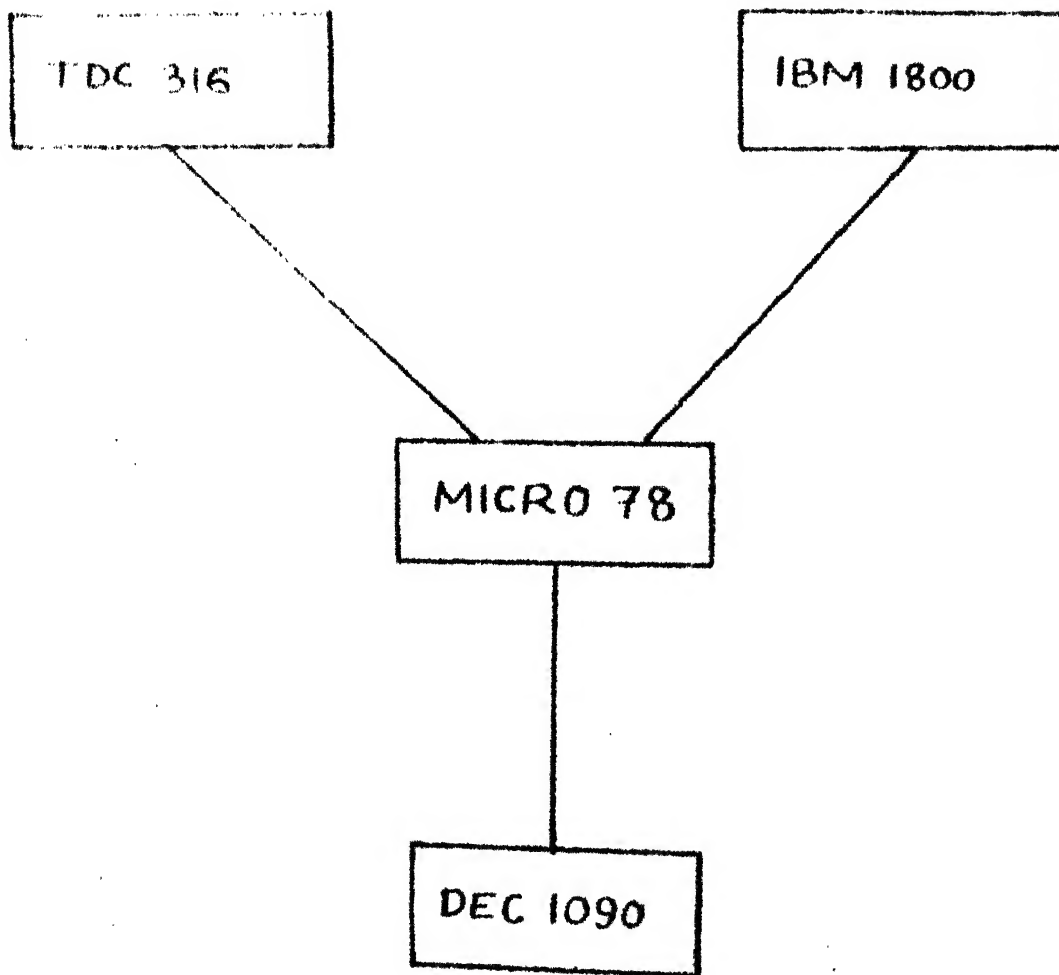


FIG. 11 : CONFIGURATION OF
IITK COMPUTER NETWORK

In the first phase, hardware interfaces were designed for TDC-316, IBM-1800 and MICRO-78 (References 1,5,6). In the second phase, i.e., the present one, the following work has been done:

- (i) Designing the hardware interfaces for MICRO-78 to DEC-1090 link.
- (ii) Making some changes in the earlier hardware interfaces.
- (iii) Implementing a part of the network software on IBM-1800 and MICRO-78.

Since the DEC-system has already got network facility, our network has been tailored to the DEC specifications. We shall now give a brief description of the DEC network architecture.

1-2. DIGITAL NETWORK ARCHITECTURE

DIGITAL provide networking facilities for their computers and the family of products that create this network is called DECNET. All implementations of DECNET adhere to a common network architecture that defines the structure and protocols used to communicate through the network. This architecture provides a modular design for DECNET and has three distinct layers:

- (i) The physical link control layer.
- (ii) The network service layer
- (iii) The application layer.

Each layer has got some well defined functions and is related to the layer above it. Following is a brief description of these layers.

(i) The Physical Link Control Layer: This is the lowest level layer and lies immediately above the physical transmission of bits over the data link. The functions of this layer are performed by the Digital Data Communications Message Protocol (DDCMP). This protocol is responsible for the correct sequencing and integrity of the data transmitted over a physical link.

(ii) The Network Services Layer: This layer performs the following functions:

- (a) Routing the messages between source and destination nodes.
- (b) Managing logical data channels, i.e., establishing logical communication links between different processes (e.g., ~~user~~ programs) so that they can exchange information regardless of their physical locations within the network.

The protocol which performs these functions is called Network Services Protocol (NSP).

(iii) The Application Layer: The functions of this layer are performed by the Data Access Protocol (DAP) which is a ~~user~~ level protocol. Its primary purpose is to permit remote file access within the DECNET environment independently of the I/O structure of the operating ~~system~~. being accessed.

In the current phase of work, only the first layer (DDCMP) has been implemented on IBM-1800 and MICRO-78.

1-3. OVERVIEW OF THE THESIS

In Chapter 2, the changes made in the earlier hardware have been discussed. In Chapter 3, we give a detailed description of DDMP and in Chapter 4, we discuss the implementation of DDMP on IBM-1800. The last chapter contains concluding remarks and comments about further development of the network.

-

CHAPTER 2

MODIFICATIONS IN THE EARLIER HARDWARE INTERFACES

As mentioned in Chapter 1, the hardware interfaces of TDC-316, IBM-1800 and MICRO-78 were designed in the first phase of the work on the network. While working on these interfaces, certain errors were detected and some improvements were visualized. To remove these errors and to implement the improvements, certain hardware changes had to be made in these interfaces.

We shall now describe these changes one by one. For the description of earlier interfaces refer to 1,5 and 6.

(1) The IBM-1800 system has channel facilities for the input and output operations. The digital output channel can be used when we have to transmit a block of characters. The words are successively transferred from the main memory to one of the digital output groups on the system, from where these words are read in and transmitted by the transmitter interface. When this channel operation is going on we cannot output any other information on any other digital/analog output group.

There are two more digital output groups which have been used for writing in the control and status registers of the receiver and transmitter respectively. The DAO group for the receiver interface has RX MODE bit and SF bit, which need to be cleared when one packet has been received. The DAO register for the transmitter has TX MODE bit on it which is set by the processor to initiate the transmission of a block of characters and the same TX MODE bit is to be cleared when the transmission of the block of characters is over.

Thus besides the transfer of words from the memory using channel operation to one of the digital output groups, the DAO is required to clear MOIE bit and SF bit of receiver control and status register, and also to set and clear the MODE bit of transmitter control and status register.

Thus, as is clear from the above paragraphs, a problem would arise when DAO channel is busy transferring a block of characters from the memory while the receiver has finished receiving the characters and thus requires DAO operation to reset the RX MODE bit and SF bit.

Some of the possible solutions to this problem are:

(a) SOFTWARE MODE BIT: If the hardware mode bit (of receiver interface) is permanently reset, the receiver interface would cause an interrupt whenever a SOM is received and the SF bit is set. To achieve data transparency, we must be able to distinguish between the SOM which is at the start of the message and any other SOM character that is in the data part of the message. This can easily be done by having a software mode bit. The software mode bit is reset initially and when an interrupt due to an SOM occurs, we know that it is due to an SOM which signifies the start of a message. Now we get the software mode bit (as we would have set the hardware mode bit) and at the same time we mask the SOM interrupt level. Now we can start receiving the rest of the message. When the complete message has been received, we reset the PLSW used for the SOML interrupt, unmask the SOM interrupt level and reset the software mode bit.

This solution could not be adopted due to the following reason:

, We still need to reset the SF bit at the end of reception of every message and this problem is no different from the one which we are trying to solve.

A solution to resetting the SF bit could be, that we do not reset SF bit when we have received a message correctly, Whenever a message is received in error, we stop receiving any further messages, wait till we find DAO not busy and then reset the SF bit. But again this is not a very elegant solution.

(b) A straightforward solution would be to connect the SF and RX MODE bits of receiving interface to some other signal points than the DAO register. We could not find any such points on the system.

(c) In this solution, we do not use the channel operation for the transfer of data from the memory of the system to the digital output group for transmission. Instead, we transfer one word at a time from the memory of the system to the digital output group using program control mode of operation of DAO (Ref. 4). This is achieved by interrupting the processor whenever the interface requests for the next character. DAO SYNC pulse is used to interrupt the processor (for DAO SYNC function see Ref. 4). The DAO READY is kept set all the time (for DAO READY function also see Ref. 4). The DAO is busy only for the time period during which it executes a write command on the digital output group. This allows us to use the DAO for other output operations (like setting and resetting of

MODE and SF bits of receiving interface) during the rest of the period.

The disadvantage of this solution is that the overhead for transfer of one word from the memory for transmission is equal to to the time required to execute the interrupt routine for this request.

This solution has been adopted, since we found that the overhead for the transmission of each word does not cause any serious problems and we have enough processor time for other computations. Data channel operation is still used for the received data.

(2) THE FIRST BYTE OF THE MESSAGE CAN ONLY BE AN 'SOM' CHARACTER (001₈): The ACK character (172₈) which could also be the first byte in a message is not used any more to interrupt the processor. ACK character has been dropped from the line protocol because it could not be used to any advantage on TDC-316 and MICRO-78 systems. Previously any message with ACK as first byte was taken as an acknowledgement message (or a control message) and any message with SOM as first byte was a data message. On these machines an interrupt was caused by the receiver interface whenever an ACK or SOM was detected as the first byte of a message, then the software was required to look at the first byte received and find out whether it was an ACK or SOM and take some action accordingly.

The same can be achieved if the first byte of the message can only be an SOM, and it is the subsequent bytes which distinguish

between the types of the messages. Again since we have only one byte, i.e., SOM which identifies the start of any message on the line, it makes the formatting of a message more flexible.

(3) On the IBM-1800 system, the clock recovery circuit has been replaced. Previously the circuit for clock recovery was a Phase Locked Loop (PLL) circuit, as shown in Figure 2-1(a). This circuit has been replaced by the mono-stable circuit as shown in Figure 2-1(b).

This change has been made because -

- (a) The PLL circuit that was used previously gave problems under voltage variations.
- (b) The circuit using the monostable is much simpler and we can tolerate $\pm 25\%$ variations in the output pulse width of the monostable.

(4) In MICRO-78 transmitter interface, the TX bit is now set by the interface itself as soon as the transmitter data register is written into by the processor and the processor has been relieved from this task. Figure 2-2(a) shows the generation of preset signal for TX.

Rx bit is now reset by the interface. Similarly, in the receiver interface, itself as soon as the receiver data register is read by the processor. Figure 2-2(b) shows the generation of reset signal for RX.

(5) In MICRO-78 transmitter and receiver interfaces, the interrupt requests are generated only when the corresponding interrupt flags are set. The setting of overrun flag in the transmitter or of the error flag in the receiver does not generate any interrupt now.

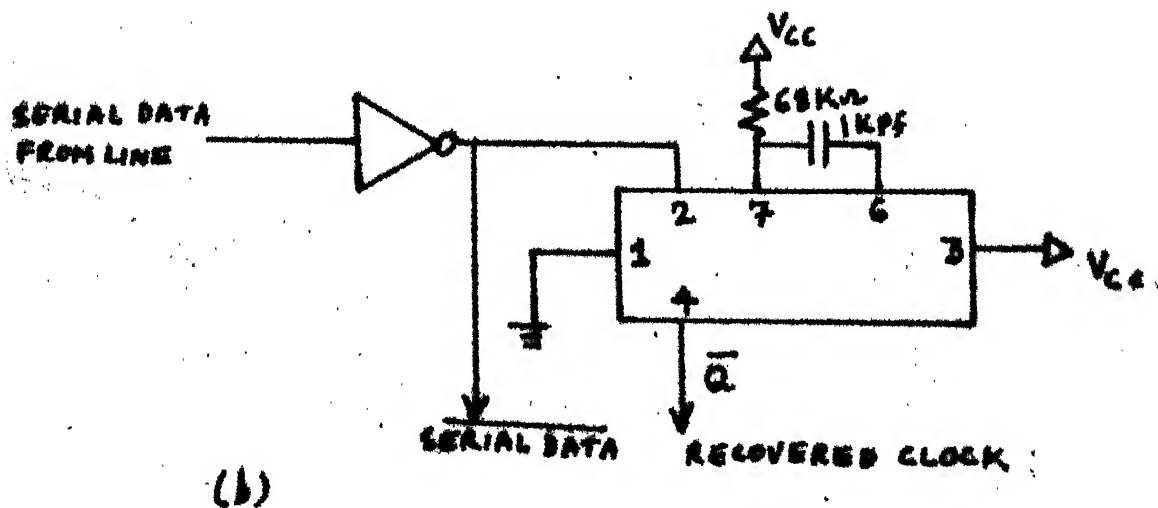
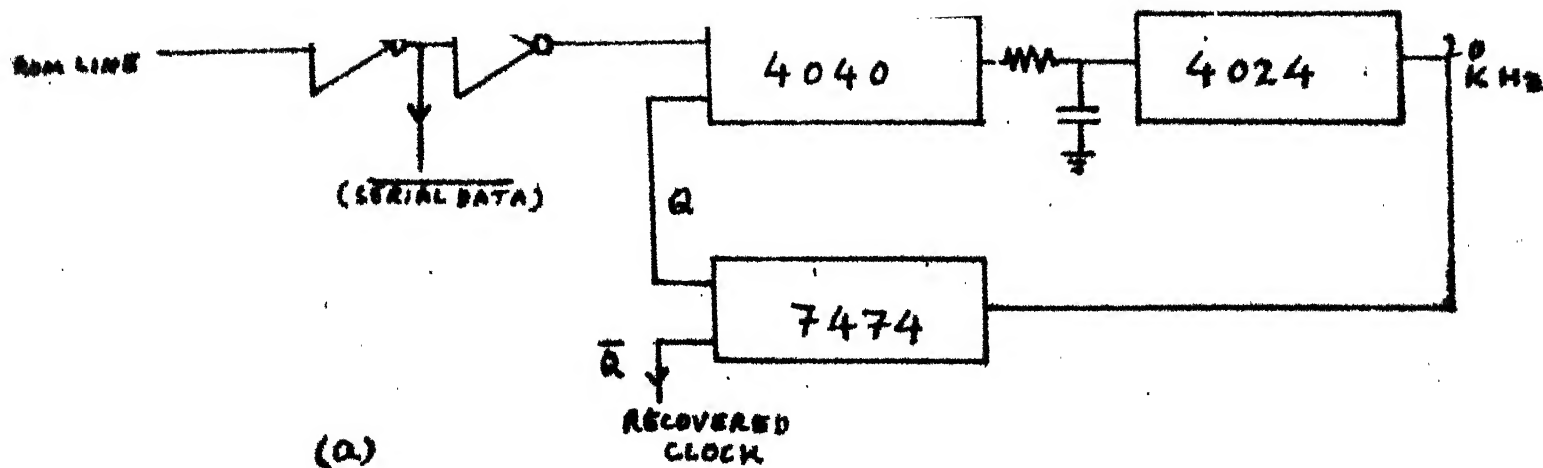


FIG. 2.1

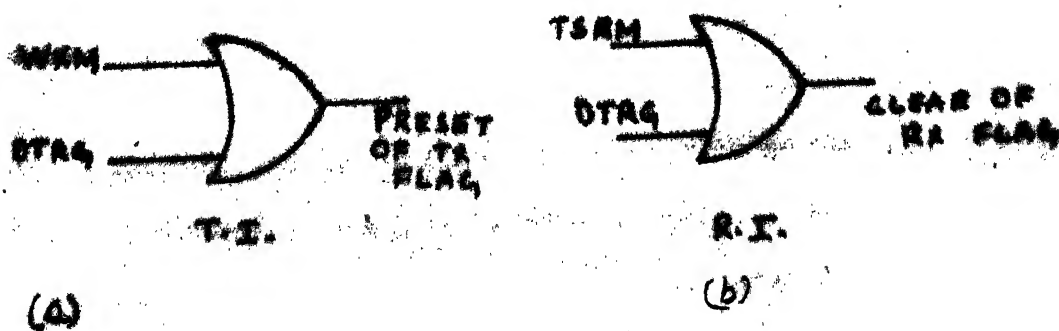


FIG. 2.2

CHAPTER 3

DESCRIPTION OF DDCMP

In Chapter 1 (Sec. 1-2), the Digital Network Architecture (DNA) was discussed and it was mentioned that DDCMP forms the lowest layer, i.e., the physical link control layer of the DNA. The main function of DDCMP is to ensure correct sequencing and integrity of the data transmitted over a physical link. This chapter discusses DDCMP in detail.

Please note that -

(i) The term 'message' used throughout is the same as the term 'packet' used in common literature.

(ii) Although DDCMP caters for both synchronous and asynchronous lines, we shall describe here only the portion suitable for synchronous links.

3-1. MESSAGE FORMATS

In DDCMP, we have three kinds of messages -

- (i) Data Messages
- (ii) Control Messages
- (iii) Maintenance Messages

Every data message carries a number (from 0 to 255) with it to enable correct sequencing at the receiving end. The numbering begins with number 1 after initialization (the details of initialization are described in Section 3-3) and is incremented by one (modulo 256) for each subsequent data message. The receiving station acknowledges the correct receipt of data messages by sending an ACK (a control message) or by sending this information along with a data message going towards that station (a piggybacked ACK). The

acknowledgement carries a number 'n' which indicates correct reception of all messages upto & including data message number 'n'. If an acknowledgement is not received within a certain time, a timeout is said to have occurred and a control message REP is sent to enquire about the status of the sent data message. Reply to REP is either an ACK or a NAK (negative acknowledgement). A NAK (another control message) is sent when a message in error is detected. STRT and STACK are two other control messages and are used for initialization of the protocol. Maintenance messages are used for diagnostic testing, bootstrapping, dumping and other such functions.

Now we proceed to describe the message formats in detail.

3-1.1 Data Messages: The format of a data message is shown in Figure 3-1(a)

(1) SOH: (1 byte) (Start of Header): This field contains the number 129(201_8) and it identifies a data message.

(2) COUNT (14 bits): This field specifies the number of bytes in the data field. The value '0' is not allowed.

(3) Flags (2 bits)

(i) Bit 0: The quick sync flag (Q SYNC flag): When this flag is on, it indicates to the receiver that the next message will not be contiguous to this one and resynchronization should follow this message. The function of this flag will be described in detail in Section 3-2.1.

SOH	COUNT	FLAGS	RESP	NUM	ADDR	BLKCK1	DATA	-----
-----	-------	-------	------	-----	------	--------	------	-------

a) DATA MESSAGE FORMAT

ENQ	ACKTYPE	ACKSUB	FLAGS	RESP	FILL	ADDR	BLKCK2
-----	---------	--------	-------	------	------	------	--------

b) ACK MESSAGE FORMAT

ENQ	NAKTYPE	REASON	FLAGS	RESP	FILL	ADDR	BLKCK3
-----	---------	--------	-------	------	------	------	--------

c) NAK MESSAGE FORMAT

ENQ	REPTYPE	REPSUB	FLAGS	FILL	NUM	ADDR	BLKCK3
-----	---------	--------	-------	------	-----	------	--------

d) REP MESSAGE FORMAT

ENQ	STRTYPE	STRSUB	FLAGS	FILL	FILL	ADDR	BLKCK3
-----	---------	--------	-------	------	------	------	--------

e) STRT MESSAGE FORMAT

ENQ	STKTYPE	STKSUB	FLAGS	FILL	FILL	ADDR	BLKCK3
-----	---------	--------	-------	------	------	------	--------

f) STACK MESSAGE FORMAT

DLE	COUNT	FLAGS	FILL	FILL	ADDR	BLKCK1	DATA	BLKCK2
-----	-------	-------	------	------	------	--------	------	--------

g) MAINTENANCE MESSAGE FORMAT

FIG. 3.1 : MESSAGE FORMATS

- (ii) Bit 1: The SELECT flag. This is used to control transmission ownership on multipoint and half duplex links. In our implementation since we have only full duplex point to point lines, we will not be concerned with this flag.

Note: COUNT and FLAGS form a 2-byte quantity. The first byte contains the 8 low-order bits of COUNT. The second byte contains the SELECT flag in the most significant bit, the QSYNC flag in the next bit and the 6 higher-order bits of COUNT in the remaining 6 bits.

(4) RESP (1 byte): This is the piggybacked ACK field and carries the number of the last consecutive correct message received from the other end by the station transmitting this message. It implies that all unacknowledged messages between the one acknowledged in the last RESP field and the one acknowledged by this RESP field (modulo 256) have been received correctly.

(5) NUM (1 byte): This field carries the number of this data message.

(6) ADDR (1 byte): This is of relevance only on multipoint links and carries the address of the tributary stations on multipoint links. This field is of no concern to our implementation.

(7) BLKCK1 (2 bytes): These 2 bytes carry the block check on the header (SOM through ADDR) using the CRC-16 polynomial $x^{16} + x^{15} + x^2 + 1$. The block check is transmitted x^{15} bit first. On reception the CRC computation should yield a zero remainder if no errors exist.

(8) Data("COUNT" No. of Bytes): This is the data field and carries as many bytes as specified in the COUNT field. This field is totally transparent to the protocol and has no restrictions on

bit patterns, groupings or interpretations.

(9) BLKCK2 (2 bytes): This field contains the block check on the data using the same polynomial as for BLKCK1.

3-1.2 Control Messages: There are 5 types of control messages: ACK, NAK, REP, STRT and STACK. We shall describe them one by one:

(1) ACK: As already mentioned, ACK message acknowledges the correct receipt of data messages. It conveys the same information as the RESP field in a data message and is used when acknowledgements are required but there is no data message to be sent in the reverse direction. Its format is shown in Figure 3-1(b).

(1) ENQ (1 byte): This field is the identifier of a control message and has value 005.

(2) ACKTYPE (1 byte): This identifies the type ACK and has value 1.

(3) ACKSUB (6 bits): This field has a value of '0'.

(4) FLAGS (2 bits): These flags are the QSYNC and SELECT flags and have the same function as in data messages.

(5) RESP (1 byte): This field has the same function as the RESP field of data messages.

(6) FILL (1 byte): This byte has value '0'.

(7) ADDR (1 byte): Same as the ADDR field of data messages.

(8) BLKCK3 (2 bytes): Block check on the control message (on ENQ through ADDR). Similar to BLKCK1 for a data message.

(ii) NAK (Negative Acknowledgement): This message is sent to the other end when an erroneous message is received from that side. The reason for error is also sent. It also includes the same

information as carried by an ACK message and thereby serves two functions: acknowledging previously received correct messages and notifying some error condition. The format for this message is shown in Figure 3-1(c). The ENQ, FLAGS, FILL, ADDR and BLKCK3 fields are the same as in an ACK message.

(1) The field NAKTYPE (1 byte): indicates the type NAK and has a value of 2.

(2) The field REASON (6 bits) identifies the source and reason for error.

The various values and the corresponding reasons are listed below:

<u>Value</u>	<u>Reasons</u>
1	Header block check error
2	Data field block check error
3	REP response
8	Buffer Temporarily unavailable
9	Receiver overrun
16	Message too long
17	Message header format error.

(3) The RESP field is the same as the RESP field of ACK and data messages, and in NAK it usually implies some error in the message with number RESP+1 (mod. 256) or beyond.

(iii) REP (Reply to Message Number): This message is sent to enquire about the status of a previously sent message and is usually sent when an ACK or NAK is not received within a timeout period. The reply to a REP is an ACK or a NAK depending on whether the receiving station has received all messages previously sent. The format of a REP is shown in Figure 3-1(d). The ENQ, FLAGS, ADDR and BLKCK3 are the same as in other control messages.

(1) REPTYPE (1 byte): This field indicates the type REP and has a value of 3.

(2) REPSUB (6 bits): This field has a value '0'.

(3) NUM (1 byte): This is the number of the last sequential numbered data message (not including retransmissions) sent by the transmitting station. This is compared against the number of the last sequential message received by the receiving station and results in an ACK if they agree and in a NAK otherwise.

(iv) STRT (Start Message): This message along with STACK is used for protocol initialization and its function will be described in detail under 'start up procedure' in Section 3-3.

Its format is shown in Figure 3-1(e). The ENQ, ADDR and BLKCK3 fields are the same as in other control messages.

(1) STRTTYPE (1 byte): This indicates the type STRT and has a value 6.

(2) STRTSUB (6 bits): This field has a value '0'.

(3) Flags (2 bits): These flags have the same function as in other messages but both flags are ones for STRT message.

(4) FILL: These 2 FILL bytes have a value of '0'.

(v) STACK (Start Acknowledgement Message): This message is sent as a response to STRT during initialization. Its format is shown in Figure 3-1(f). ENQ, FLAGS, FILL, FILL, ADDR and BLKCK3 fields are same as in STRT.

(1) STCKTYPE (1 byte): Indicates STACK type and has value '7'.

(2) STCKSUB (6 bits): Has a value of '0'.

3-1.3 Maintenance Messages: The DDCMP protocol operates in two basic modes: (1) On line or normal running mode, (2) Off line or the maintenance mode. The previous messages correspond to the on line mode. The maintenance mode may be used for basic diagnostic testing and simple operating procedures such as bootstrapping, down line loading and dumping. It provides a basic envelope compatible with DDCMP framing, link management and CRC check for bit errors but does not include any error recovery, time outs or sequence checks.

The format of a maintenance message is shown in Figure 3-1(g).

(1) DLE (1 byte): This field identifies a maintenance message and has the value 144(220₈).

(2) COUNT (14 bits): This field specifies the number of 8 bit bytes in the DATA field. The value '0' is not allowed.

(3) FLAGS (2 bits): Both these flags have value '1'.

(4) FILL: These 2 fill bytes have a value of '0'.

(5) ADDR (1 byte): Same as for other messages.

(6) BLKCK1 (2 bytes): CRC check bytes for the header.

(7) DATA ('COUNT' No. of bytes): This is the data field and contains number of data bytes specified in the COUNT field.

(8) BLKCK2 (2 bytes): CRC check on the data

3-2. OPERATION OF DDCMP

The functions of DDCMP can be grouped under three heads:

- (1) Framing
- (2) Link Management
- (3) Message Exchange

These are discussed in the following subsections.

3-2.1 FRAMING: Framing is the process of locating the beginning and end of a message at the receiving end. For this, the receiving end must synchronize at the bit, byte and message levels. We shall describe how DDCMP accomplishes these functions.

(1) Bit Synchronization: This function is achieved by the hardware interfaces (or modems) and is not a part of DDCMP.

(2) Byte Synchronization: Data is sent over the line as a sequence of bits grouped into 8-bit bytes. The receiver must be able to locate the proper 8 bit window in the bit stream and stay in step with it for every subsequent 8 bit grouping. On a synchronous line, this function is accomplished by searching for a unique byte called a SYN byte (value 226_8). Once this is found every subsequent group of 8 bits forms a byte.

DDCMP specifies that the receiver must locate and lock on to 2 consecutive SYN bytes to achieve byte synchronization. The transmitter must send four or more SYN bytes to allow for the loss from missynchronization and hardware interface constraints.

After the reception of a message, two possibilities exist:

(i) The next message immediately follows the current message. In this case the receiver will remain synchronized and reception next of the message will continue.

(ii) The next message does not immediately follow this message. In this case, byte synchronization is assumed to be lost and the receiver must resynchronize for receiving the next message. Hence the next message must be preceded by a synchronization sequence (i.e., 4 or more SYN bytes).

On some communication interfaces like the DMA type, the received data may be buffered in the device and by the time the software driver comes to know that the next message is not contiguous, the device may have buffered many bytes further ahead on the link. So, a long synchronization sequence is required in such cases to account for potential buffering in the interface. The length of the SYN sequence should be 4 more than the number of bytes buffered. A value of 8 is generally suitable.

To reduce the length of the synchronization sequence, the QSYNC flag is used (This is one of the link flags described in Section 3-1.1). When this flag is '1', it notifies the receiver that the next message will not be contiguous to the current one and the synchronization sequence preceding that message may be the short sequence. So, on seeing a QSYNC flag the software driver at the receiver can start resynchronizing immediately after the current message without looking ahead into the next message, thus enabling the receiver to synchronize on a short sequence.

(3) Message Synchronization: The beginning of a message is located by searching for one of the three bytes SOH, ENQ and DLE (which are the starting bytes of data, control and maintenance messages respectively, as described in Section 3-1). One of these bytes must appear immediately after the SYN sequence or immediately after the previous message's last byte. Otherwise, the byte synchronization is assumed to be lost. After detecting the first byte, the end of the message is determined by the following rules:

(1) If the starting byte is SOH or DLE then the next 5 bytes will complete the message header, followed by 2 bytes of CRC check on header followed by 'COUNT' number of bytes of data followed by 2 bytes of data block check. (See Section 3-1).

(2) If the starting byte is ENQ then the next 5 bytes will complete the message followed by 2 bytes of CRC check.

Since no pattern searching is done once the starting byte is found, the data field is totally transparent.

The receiver tries to achieve message synchronization only under the following conditions:

- (1) Initially on start up
- (2) If messages are not contiguous
- (3) If the QSYNC flag is set in the current message, resynchronization is done at the end of this message.
- (4) If CRC error or other error occurs that might have caused synchronization to be lost.

In a simple implementation, synchronization may be done after every message, but this is somewhat less efficient.

3-2.2 Link Management: The SELECT flag in all the messages is meant for link management, i.e., controlling the transmission ownership on half duplex and multipoint lines. However, in the case of full duplex point to point lines, there is no link management required. In this case the address field is kept '1' and the SELECT flag has no role to play. We shall not discuss this component of DDCMP since in our implementation only point to point full duplex links are involved.

3-2.3 Message Exchange: This component of DDCMP caters for the exchange of error-free and correctly sequenced data messages. The following is a description of this component of DDCMP.

(1) The transmitter assigns the next sequential message number n to the data message, adds a CRC block check and sends it. After sending a message, a timer is started.

(2) The receiver, after receiving the message, checks it for any errors and compares the message number with the next expected and takes the following actions:

- (i) If the number is correct a positive acknowledgement (ACK) carrying this number is sent back and the next expected number is incremented modulo 256.
 - (ii) If the message is in error, a negative acknowledgement with error reason is sent. The NAK also carries with it the number of the last correctly sequenced message received. NAK is sent to hasten the process of re-transmission, without waiting for a time out. A NAK not only indicates an error, but also acknowledges the correct reception of messages upto the number carried by it.
 - (iii) If the message is correct but its number is not equal to the expected message number, it is simply ignored.
- (3) The transmitter follows the following procedure:
- (i) If it receives a positive acknowledgement, it compares its number with the one expected. If it agrees, the message is released, the user is notified and the timer is stopped. If the acknowledgement number does not agree with the expected number, it is ignored.
 - (ii) If it receives nothing and a timer expires, it sends an enquiry message called REP to the receiver with the number of the message previously sent. The response to a REP is an ACK or a NAK depending on whether the message with that number was correctly received or not. The transmitter timer is again started after sending a REP and if it expires again, this process is repeated.

After some specified number of timeouts, the user is informed, who may decide that the link is out of service.

The following points may be noted about this procedure:

(1) The transmitter can send upto 255 messages without waiting for their ACKs. An ACK confirms the correct reception of all messages between the last acknowledged message and the one with the number contained in this ACK. Same is the case for a NAK. If an ACK gets corrupted, the information lost in it is automatically included in a subsequent ACK, thus eliminating the sending of REP messages if the ACKs are received prior to the expiration of the transmitter timer.

(2) An ACK need not be sent separately, but can be piggybacked with a message going in that direction (the RESP field contains the ACK number).

3-2.4 Message Exchange Variables: The following variables are used in the message exchange state tables. Arithmetic operations and comparisons for these variables are always modulo 256.

- (1) N: It is the number of the highest sequential data message transmitted by this station. It is sent in the NUM field of REP messages.
- (2) R: The number of the highest sequential data message received at this station. Sent in the RESP field of data messages, ACKs and NAKs.
- (3) A: The number of the highest sequential data message that has been acknowledged to this station.
- (4) T: It is the number of the next data message to be transmitted. When sending a new data message, T will have the value N+1. When retransmitting, T will be set to A+1 and will advance to N+1.
- (5) X: The number of the last data message that has completed transmission.

3-2.5 Control Flag Variables: There are three control flags which specify the sending of control messages.

(1) SACK (Send ACK Flag): This flag is set when either R is incremented (i.e., a new data message has been received) or a REP message is received which requires an ACK reply. It is cleared when a DATA message is sent with a piggybacked ACK, an ACK message is sent or the SNACK flag is set.

(2) SNACK (Send NAK flag): This flag is set when an erroneous message requiring a NAK reply is received. It is cleared when a NAK is sent or when SACK is set.

(3) REP (Send REP flag): This flag is set when a reply timer expires in the running state indicating that a REP should be sent.

3-3. DDCMP STATES AND THE START UP PROCEDURE

DDCMP can be in one of the following states:

(1) HALTED: This state signifies that the protocol is not running.

(2) STARTING: This means that an attempt is being made to initialize the protocol. This state has two internal states, ISTRT and ASTRT, the details of which will be soon made clear.

(3) RUNNING: This is the on line state ^{of} DDCMP in which numbered data messages are exchanged.

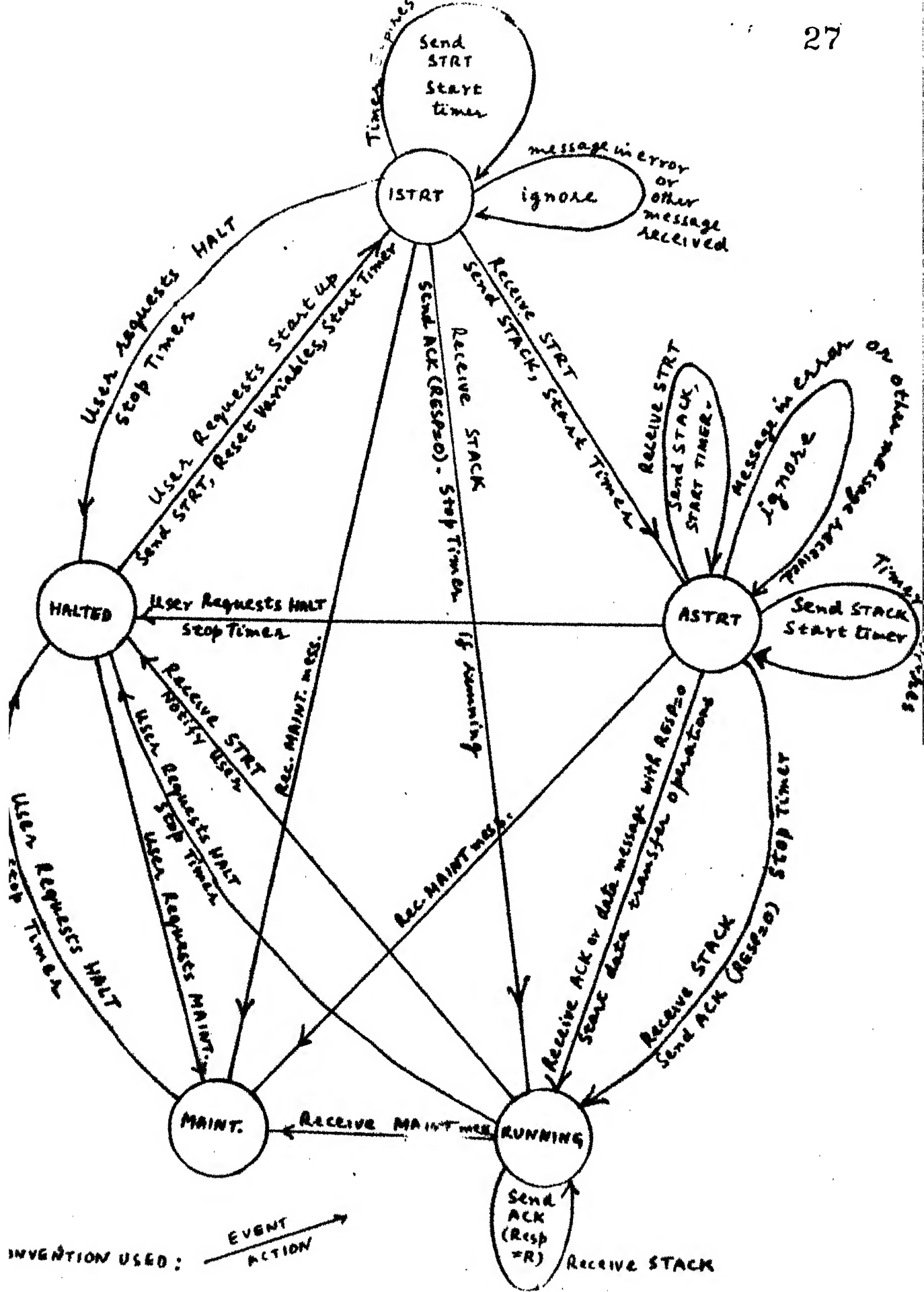
(4) MAINTENANCE: This is the off line state of DDCMP in which maintenance messages are exchanged.

From the halted state, protocol initialization begins when a start up command is given by the user. An exchange of STRT-STACK messages takes place and state transitions take place from HALTED to RUNNING via ISTRT and ASTRT. These transitions are depicted in the state diagrams shown in Figure 3-2.

3-4. THE RUNNING STATE AND DATA TRANSFER

It is the RUNNING state in which numbered data messages are exchanged, acknowledgements sent, retransmissions done and all similar actions taken. Following is a table showing various events and corresponding actions in the RUNNING state. Note that all comparisons are modulo 256. The priority of transmission is given as: NAK, REP, data message, ACK.

<u>Event</u>	<u>Action</u>
.Receive data message ($NUM=R+1$)	Give message to user, set SACK flag.
.Receive data message ($NUM \neq R+1$)	Ignore
.Receive message with errors	SET SNAK flag with appropriate reason
.Receive REP ($NUM=R$)	SET SACK
.Receive REP ($NUM \neq R$)	SET SNAK, reason 3.
.Receive ACK or data message with $A \leftarrow RESP$, $A \leq N$	For all messages ($A \leq NUM \leq RESP$), notify user of their receipt and release them. $A \leftarrow RESP$ If $T \leq A$, then $T \leftarrow A+1$ If $A \leq X$, start timer If $A > X$, stop timer
.Receive ACK or data message with $RESP < A$ or $RESP > N$	Ignore
.Receive NAK with $A \leq Resp \leq N$	For all messages with $A \leq NUM \leq Resp$ inform user of their reception, $A \leftarrow Resp$ $T \leftarrow A+1$ Stop timer



- .Receive NAK with RESP. $\leftarrow A$ Ignore
or Resp $> N$
- .Timer expires Set SREP flag
- .Transmitter idle, SNAK set Send NAK, Clear SNAK.
- .Transmitter idle, SNAK clear, SREP set Send REP, clear SREP.
- .Transmitter idle, SNAK and SREP clear $T \leftarrow N+1$ Retransmit message T, $T \leftarrow T+1$,
Clear SACK.
- .User requests message to be sent, $T=N+1$, Transmitter idle, SNAK and SREP clear Send message with number $N+1$, $N \leftarrow N+1$,
 $T \leftarrow N+1$
Clear SACK
- .Transmitter idle, NAK and SREP clear, No user request waiting, SACK set Send ACK
Clear SACK.
- .Data Message (NUM) transmission completed on link $X \leftarrow \text{NUM}$
If $A < X$ and timer not running start timer
If $A \geq X$, stop timer
- .REP message transmission completed on link Start timer

Note: (1) The term 'send' used above means put the message in a transmitter queue.

(2) The term 'transmitter idle' above means that queuing space for transmission is available.

CHAPTER 4

IMPLEMENTATION OF DDCMP ON IBM 1800

In the last chapter, we discussed in detail the functions and specifications of DDCMP. In this chapter, we will describe how DDCMP has been implemented on IBM-1800.

4-1. BASIC STRUCTURE

ON IBM 1800, we have got 6 basic routines, viz., the transmitter interrupt routine, the receiver interrupt routine, the DDCMP receive routine (DDR), the user transmit routine (USEND) and the user receiver (USRCV) routine. These routines are interfaced by various queues as shown in Figure 4-1. The functions of each of these routines are briefly described below:

(1) USEND: This is the user's routine and is used to pass on messages and commands to DDCMP. Whenever the user wants to send a message or give some command (startup, halt, etc.) he takes a free buffer from FREEQ, which is a pool (queue) of free buffers, puts his message/command in that buffer and places it in a queue, Q1.

(2) DDT: This routine is a component of DDCMP that is responsible for the transmission aspect. This routine decides what has to be transmitted next, i.e., whether a control message a new data message or an old data message should be sent next. It also checks up whether there is any halt or startup message in Q1. Accordingly, it takes appropriate action, i.e., halting the protocol or sending the next message after proper formatting of header, CRC, etc. The messages to be sent are placed in a queue Q2.

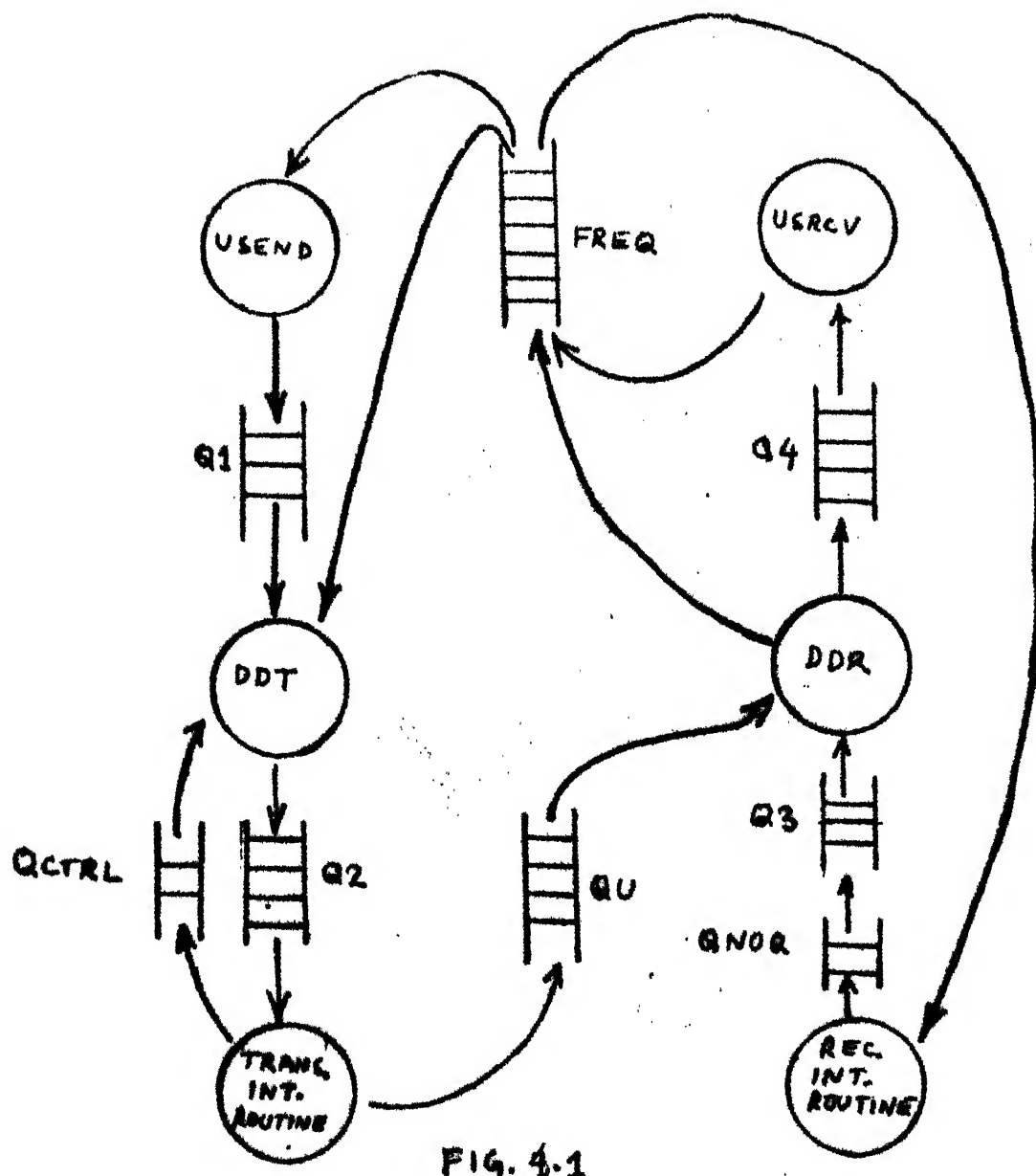


FIG. 4.1

NOTE:

- I) THE ARROWS INDICATE VARIOUS PATHS THAT MAY BE TAKEN BY THE BUFFERS WHEN DDMP IS RUNNING.
- II) QCTRL HAS ATMOST ONE BUFFER IN IT AND IS MEANT TO KEEP A BUFFER FOR SENDING CONTROL MESSAGES.
- III) QNOB HAS ALWAYS GOT ONE BUFFER FOR RECEIVING A MESSAGE.

(3) Transmitter Interrupt Routine: This routine takes messages one by one from Q2 and sends them on the link. After a complete message has been sent, this routine also takes some other actions like starting a timer, stopping a timer etc. If a control message has been sent, its buffer is returned to ~~Q2~~ Q2, but if a data message has been sent, it is placed in QU, the queue of unacknowledged messages.

(4) Receiver Interrupt Routine: This routine is responsible for receiving messages from the line and placing them in the queue Q3. It basically performs the function of message framing.

(5) DDR: This routine is the receiving component of DDCMP. It takes messages from Q3, examines them, checks their CRC block checks and accordingly takes appropriate actions. The correct data messages are placed in a queue Q4 for the user.

(6) USRCV: This is a user's routine that takes messages one by one from Q4 and uses them appropriately. The freed buffers are sent to FREQ.

4-2. SPECIAL FEATURES OF THE IMPLEMENTATION

Due to certain hardware peculiarities of the IBM-1800 and the hardware interface designed for it, certain special features had to be incorporated in the implementation of DDCMP. They are described below:

(1) Although DDCMP specifies that the receiving end should synchronize only after the reception of 2 consecutive SYN characters, our interface synchronizes after receiving just 1 SYN character. Moreover, the value of the SYN character is 026 instead of 226 as specified by the protocol.

LIBRARY
CENTRAL LIBRARY

(2) The hardware interfaces between MICRO-78 and IBM-1800 are designed to recognize the character 001_8 as the starting character of any message and an interrupt is generated when this character is received. However, in DDCMP we have 3 starting characters $SOM(0201_8)$, $ENQ(005_8)$ and $DLE(220_8)$. So, we precede every message with the character 001_8 . This satisfies the purpose of implementing DDCMP without making any hardware changes.

(3) We resynchronize after every message. At least 2 SYNs are sent before any message. Consequently, the flag QSYNC is not used in the implementation.

(4) The IBM-1800 has only word operations (each word of 16 bits) and the minimum addressable unit is also one word. In data channel operations also, the count can be given only in number of words. So, only an even number of 8 bit bytes can be sent or received. However, this does not make any difference as explained below:

- (i) If we have to send an odd number of bytes, we send an extra (dummy) byte (preferably SYN) with it, which will be ignored at the other end.
- (ii) If we have to receive an odd number of bytes, we receive an extra byte, but simply ignore it while processing the message.

4-3. BUFFER FORMATTING AND QUEUES

Buffers are arranged as doubly linked lists. Every list has a header, which occupies three words in the format shown in Figure 4-2(a). The buffers on the transmitting side have the format shown in Figure 4-2(b).

WORD 1

FORWARD POINTER

WORD 2

BACKWARD POINTER

WORD 3

NO. OF BUFFERS IN THE LIST

a) FORMAT OF A HEADER

WORD 1

FORWARD POINTER

WORD 2

BACKWARD POINTER

TYPE OF MESSAGE

MESSAGE NUMBER

COUNT

OUTPUT CHANNEL NO.

SYN

SYN

SYN

SOM

↑
MESSAGE
(INCLUDING
HEADER)
↓

b) BUFFER FORMAT
FOR TRANSMITTING
SIDE

WORD 1

FORWARD POINTER

WORD 2

BACKWARD POINTER

COUNT = 6

INPUT CHANNEL ADDR

SYN

SOM

HEADER + CRC of
HEADER
(4 WORDS)

COUNT

INPUT CHANNEL ADDR

↑
DATA AND ITS
CRC
↓

c) BUFFER FORMAT
FOR RECEIVING
SIDE

It should be noted that:

- (a) Message number is relevant only in the case of data messages.
- (b) Count initially contains the number of bytes in the data part of a data message and is later substituted by the number of words to be transmitted on the line.
- (c) The field 'output channel number' is not being used presently since we are not using the data channel operation but has been kept for use in any future changes.
- (d) The 'type' field has the following interpretation:

<u>Value</u>	<u>Meaning</u>
0	Data message
1	ACK message
2	NAK message
3	REP message
6	STRT message (Also serves the purpose of START protocol command from user)
7	STACK message
99	Halt command (from user to DDMP)

The buffers on the receiving side have the format shown in Figure 4-2(c).

Please note that:

- (a) Count 1 contains the number of words to be received initially (this includes only the header) plus one (See Ref. 4). This count is always 6.
- (b) Count 2 contains the number of words to be received in the data part plus one (Ref. 4).

4-4. DATA VARIABLES AND CONTROL FLAGS

As already described in Chapter 2, various data variables R,N,A, T,X and control flags SACK, SNAK and SREP are required for DDMP

implementation. Besides, another variable indicating the state of DDCMP is also there. All these variables are global. The variable STATE has the following interpretation:

<u>Value</u>	<u>State of DDCMP</u>
0	ISTRRT
1	ASTRT
2	RUNNING
3	HALTED
4	MAINTENANCE

4-5. ROUTINES FOR QUEUE OPERATIONS

There are three routines for various queue operations. These routines may be called by any other routine which needs to perform operations on the queues. These routines are:

- (1) SHFTQ: This routine shifts a buffer from one queue to another, i.e., it removes the first buffer of one queue and appends it to the end of another queue. Its calling sequence is

```
CALL SHFTQ
DC QFROM
DC QTO
```

The parameter QFROM is the address of the header of the queue from which a buffer has to be removed and QTO is the address of the header of the queue to which it is to be shifted. SHFTQ should be called only when QFROM is non empty, otherwise a fatal error is assumed to have occurred and the protocol halts after printing an error message (Refer Appendix II).

- (2) SRCHQ: This routine searches a list for a buffer with a particular message number i.e., a particular value in the NUM field (Refer Figure 4-2(b)) and after finding that buffer, removes it from the original list and places it at the end of another list. Its calling sequence is:

```
CALL SRCHQ
DC QFROM
DC QTO
DC NUM
```

QFROM is the address of the header of the list to be searched, QTO is the address of the header of the list to which the designated buffer is to be shifted and NUM is the address of the location containing the message number required in the search. If the search succeeds, a flag SRERR (a global variable) is set, otherwise it is reset. The calling routine should examine SRERR and take appropriate action.

- (3) EMFTQ: This routine shifts all the buffers of one queue to another queue. Its calling sequence is:

```
CALL EMFTQ
DC QFROM
DC QTO
```

QFROM and QTO explain themselves. QFROM may be empty.

4-6. DETAILED EXPLANATION OF VARIOUS ROUTINES - (1) USEND:

This routine takes an empty buffer from FREQ, places it in a temporary queue TEMQ* and fills it as follows:

- (i) If the user wants to give a command which will halt the protocol after sending all the previously given messages he places number 99 in the type field.
- (ii) If the user wants to give a startup command, he places No. 6 in the type field.
- (iii) If the user wants to send a data message, he places No. '0' in the type field, the number of data bytes in the COUNT field and the data in the data field.

*TEMQ is another queue which is used to hold a buffer temporarily while it is being processed by some routine. This queue never contains more than 1 buffer.

After the buffer has been filled with relevant information it is shifted from TEMQ to Q1.

If the user wishes to halt the protocol immediately (i.e., even before the previous messages are transmitted), he simply sets a flag called SHALT.

(2) DDT: The basic function of this routine has already been explained. The details are clear from the flowchart given in Figure 4-3(a). The following points will help in understanding the flowcharts:

- (i) There are six header routines, MES, ACK, NAK, REP, STRT and STACK (Figures 4-4(a) to (f)), which generate the headers for that message type. These routines fill in all the information shown in the transmitting side buffers (Figures 4-2(b)) except the data in data messages and the CRC check on the data. These routines need only one parameter to be passed, i.e., the address of the header of that queue which contains the buffer to be filled in at the front.
- (ii) All arithmetic and comparison operations shown are modulo 256.
- (iii) A routine for CRC generation is called at various points. The calling sequence of this routine is:

```
CALL CRCGN
  DC ADDR
  DC COUNT
  DC FLAG
```

ADDR is the address of the first location of the area containing the data for which CRC is to be generated or checked. COUNT is the address of the location containing the byte count of data. FLAG is the location containing either a 0 or 1 depending upon whether we want CRC to be generated or checked, respectively. When we call CRCGN

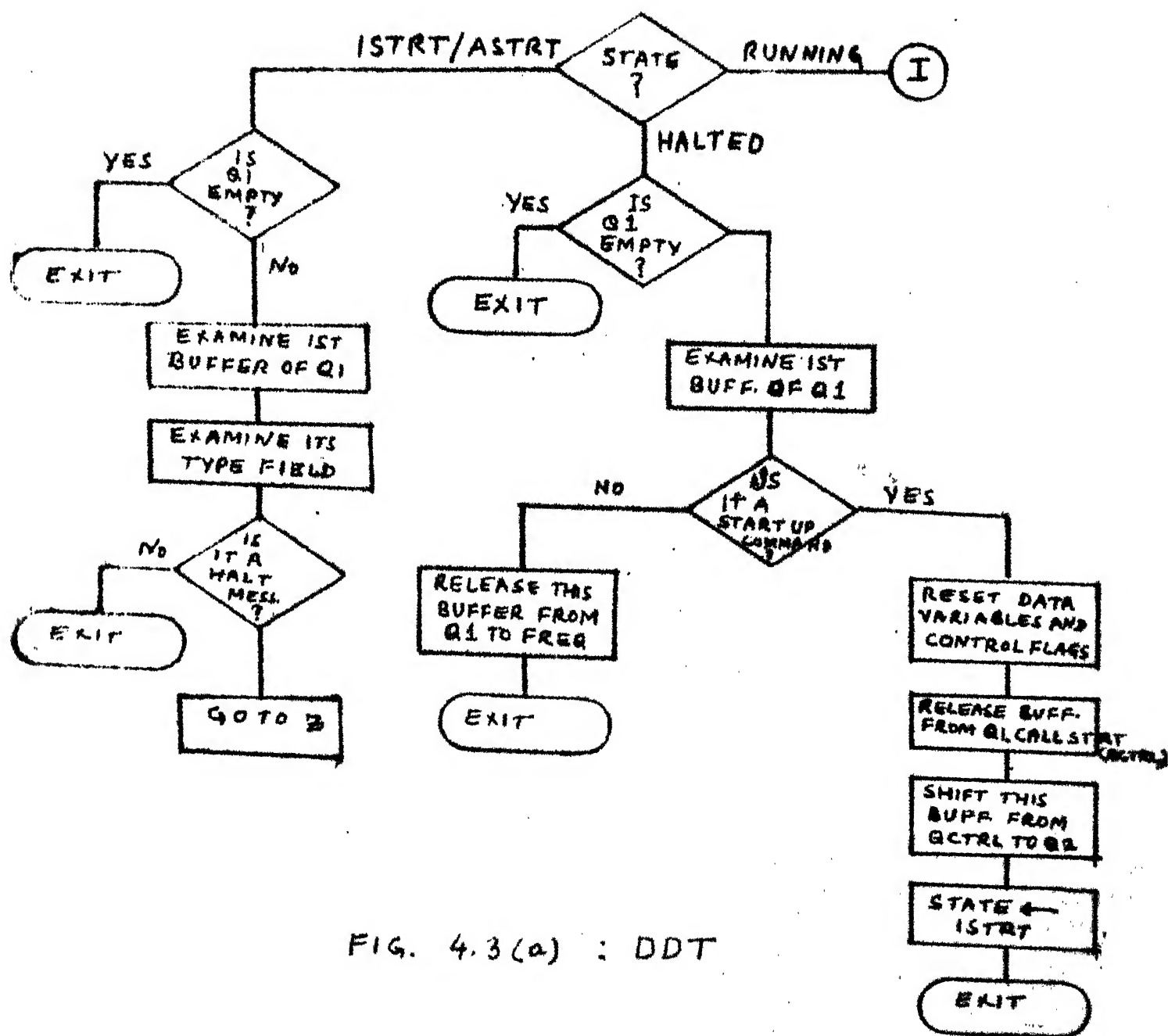


FIG. 4.3(a) : DDT

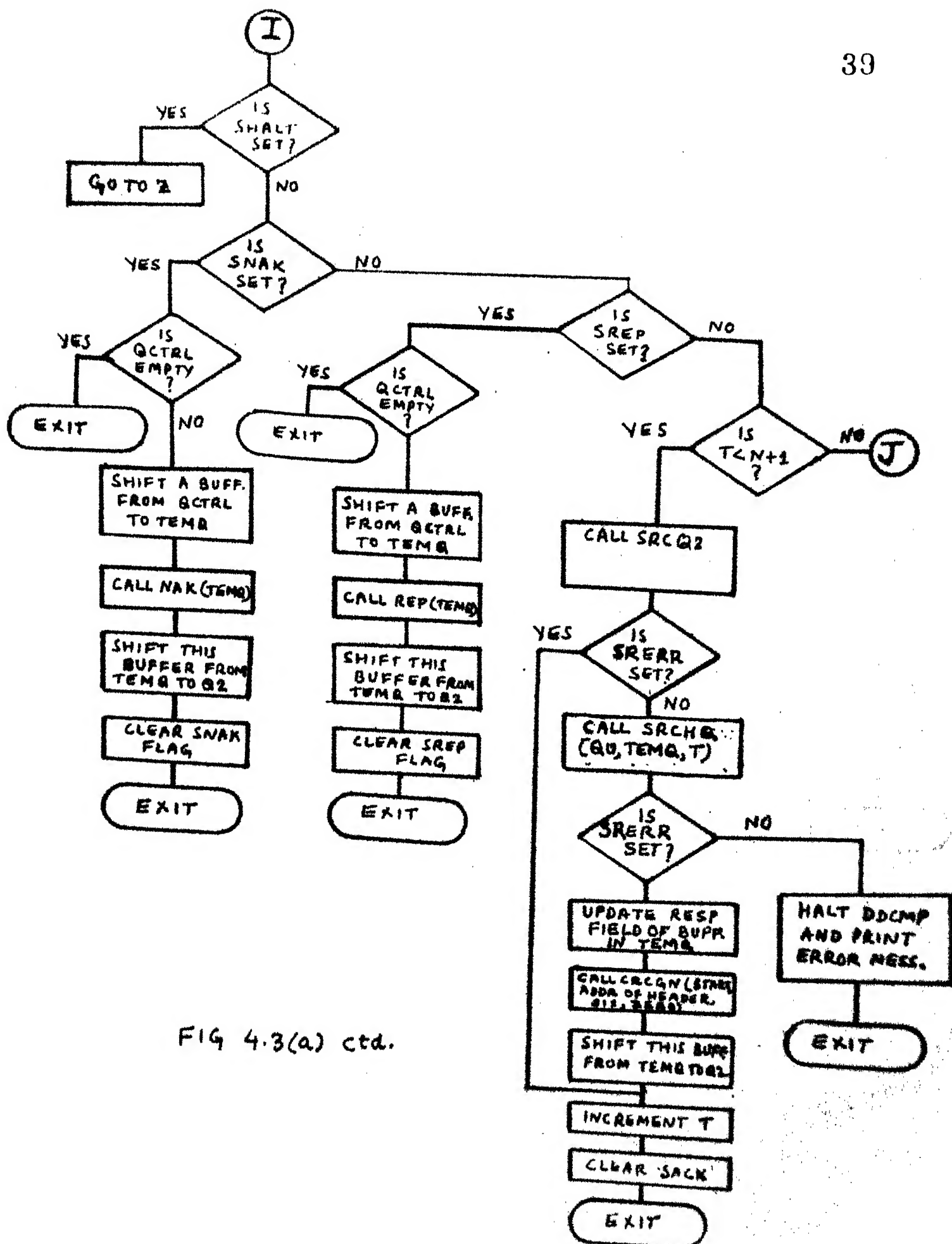


FIG 4.3(a) ctd.

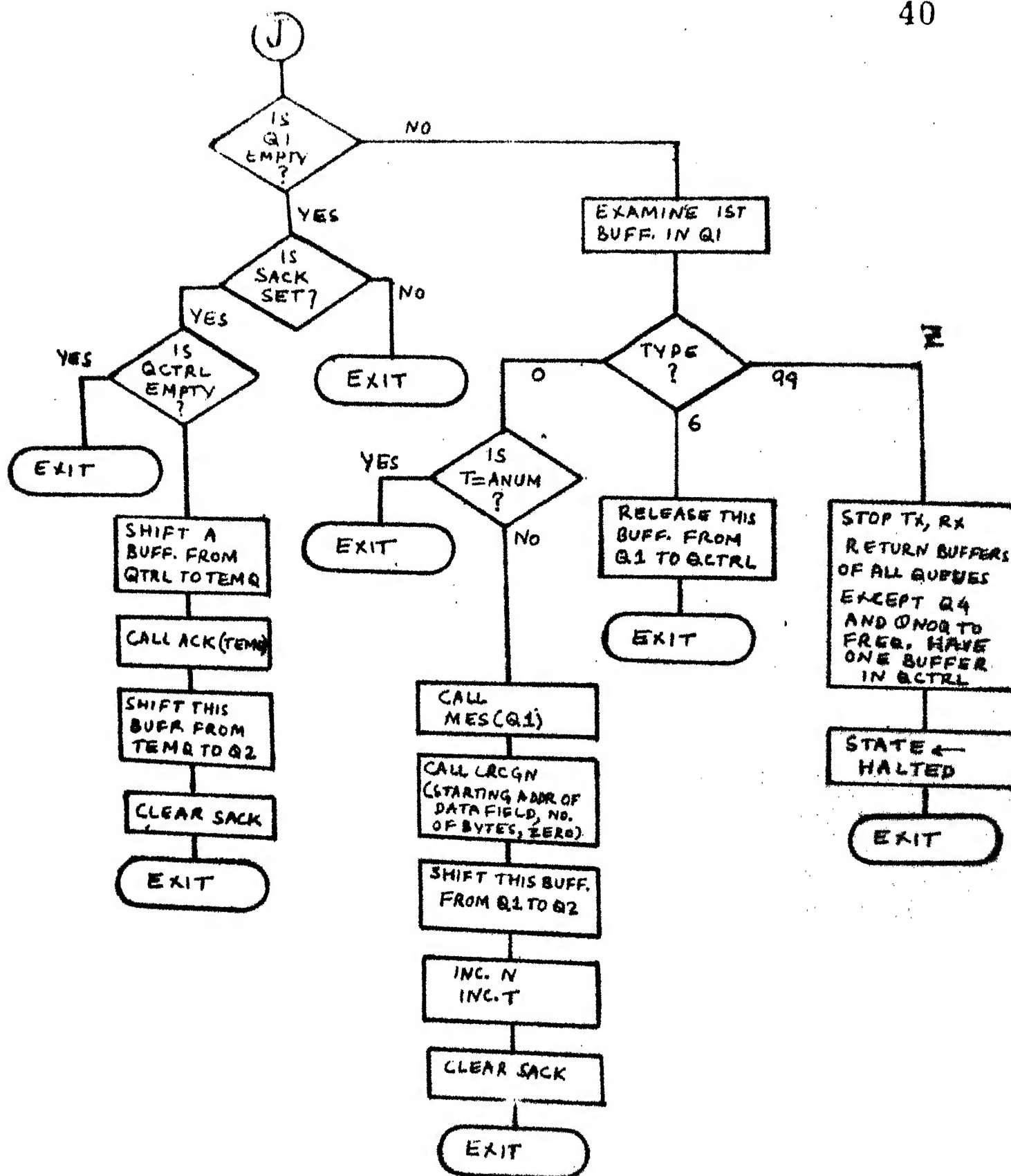
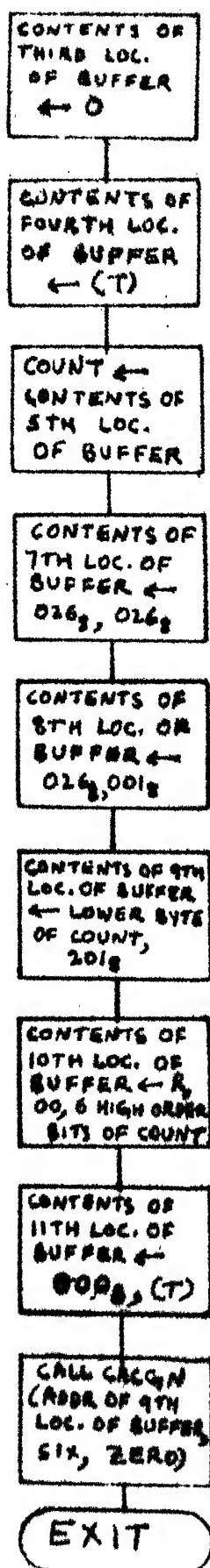
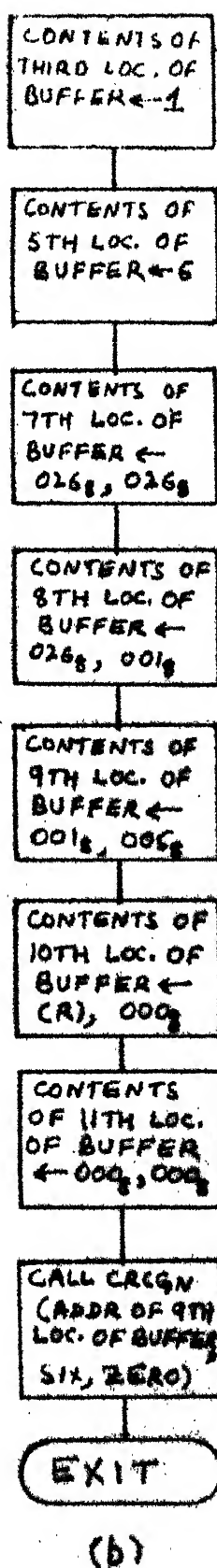


FIG 4.3(a) ctd.

MES



ACK



NAK

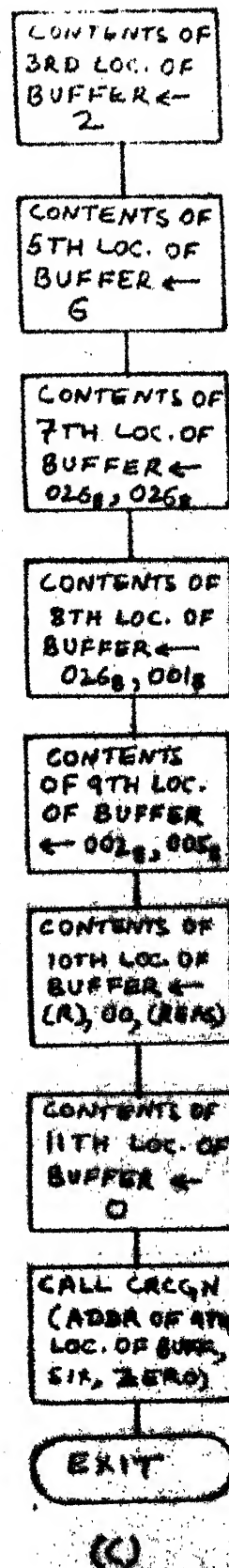
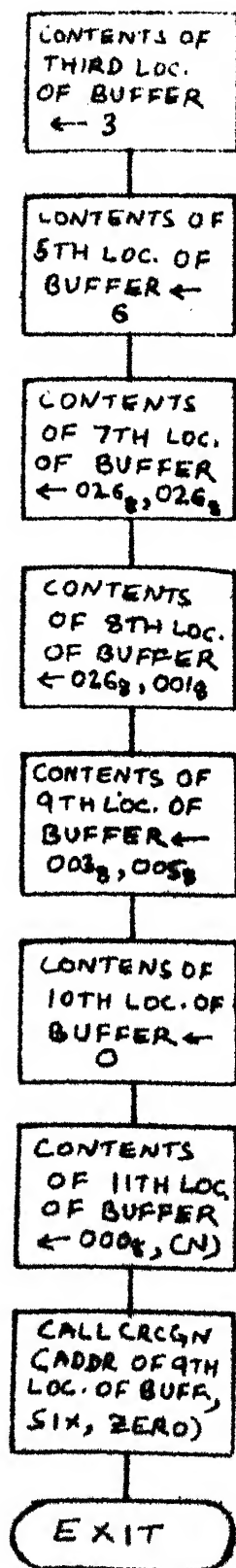


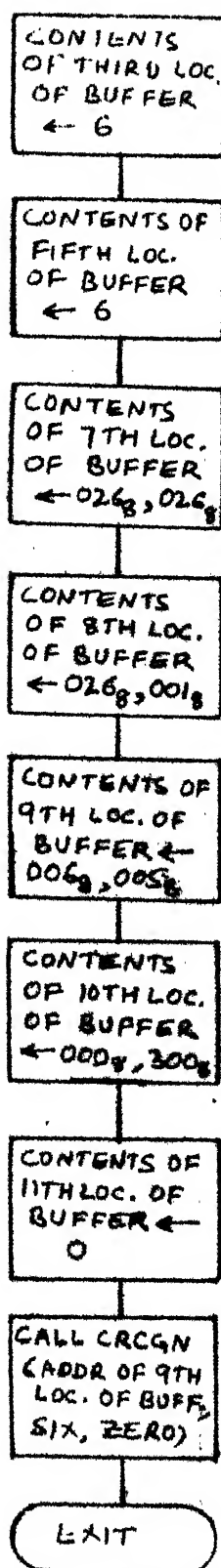
FIG 4.4

REP



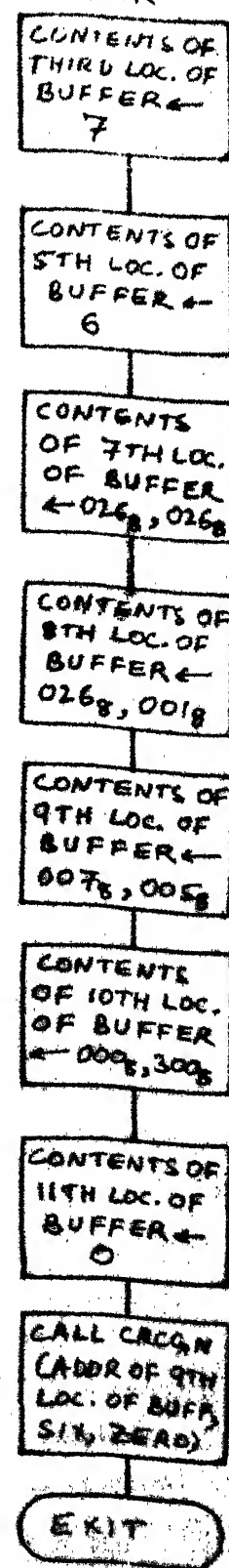
(d)

SIRT



(e)

STACK



(f)

for generating CRC (i.e., the location FLAG contains a '0') the two CRC bytes are placed in the two locations immediately following the data. When we call CRGN for checking CRC (i.e., the location FLAG contains a '1'), the location FLAG will be set to 0 if no error is there and will remain 1 otherwise.

(3) Transmitter Interrupt Routine: An interrupt is generated after each word is transmitted. The interrupt routine initiates the transmission of next word, if any. When all words have been transmitted certain actions are performed, as are clear from the flowchart (Figure 4-5).

(a)

(4) Receiver Interrupt Routines: An interrupt is generated on the receiving side when an SOM(001₈) is detected and the mode bit of the receiver is off. As a result of this control comes to the SOM interrupt routine, which performs the following functions:

- (i) It checks whether a free buffer has already been allocated for reception of a message. This is done by checking a flag called ALLOC flag. ALLOC = 0 means that a free buffer has not been allocated and ALLOC = 1 means that it has been allocated.
- (ii) If a free buffer is not already allocated, FREQ is checked and if a buffer is available, it is allocated, otherwise the previously allocated buffer (which is still in QNOQ, a queue which holds the buffer for reception of a message) is not transferred to Q3 but is retained for reception of the new message thus destroying the previous message.
- (iii) It then initiates the data channel for receiving the header and its CRC.

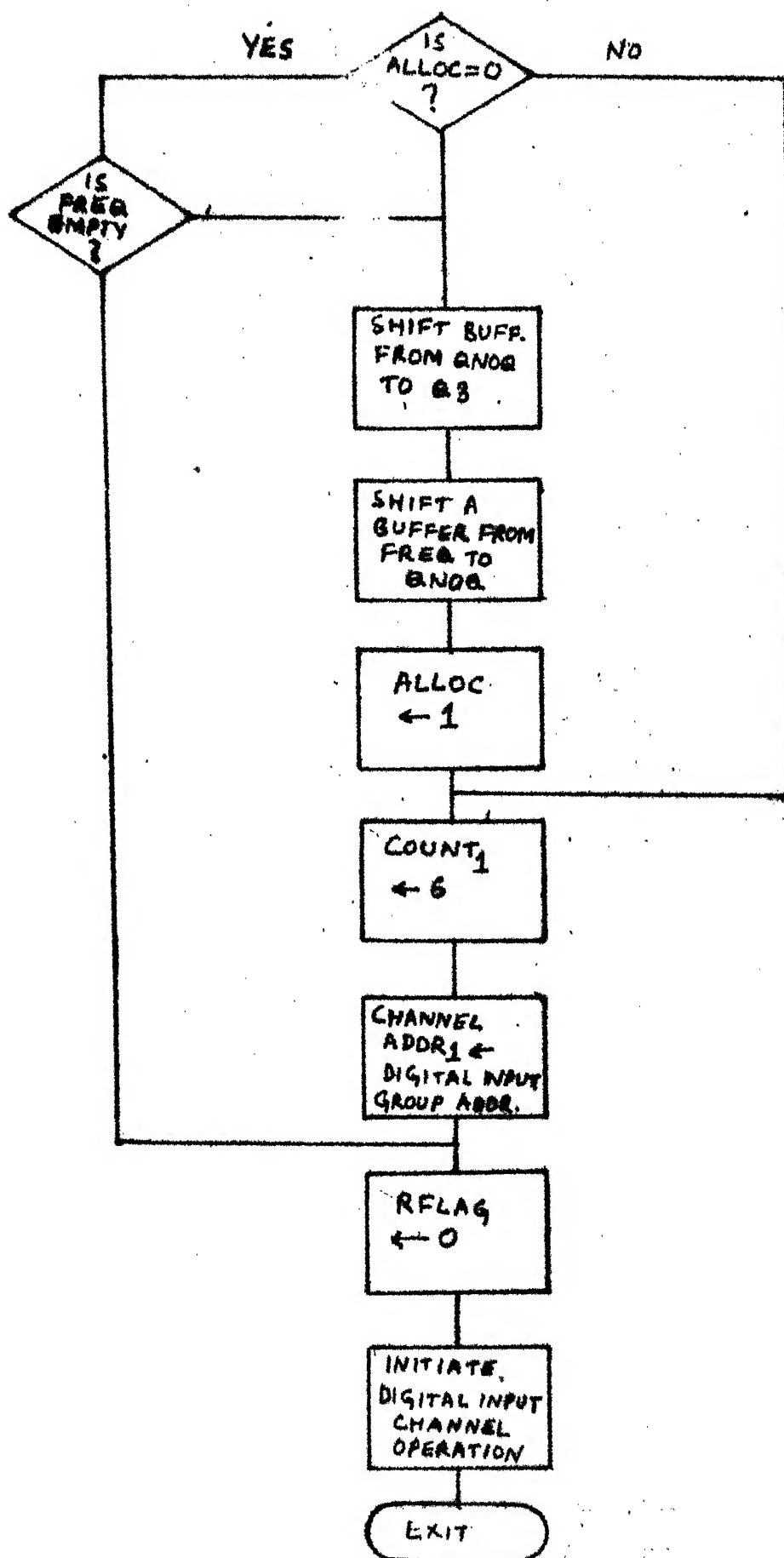


FIG. 4-5 (b) : SOM INTERRUPT ROUTINE

- (iv) It sets RFLAG to 0. RFLAG is checked in the Digital Input Channel interrupt routine to find out whether the header or the data part has been received. When the digital input channel completes receiving the specified number of words, it causes an interrupt. In its interrupt routine, the following actions are taken:

RFLAG is checked to find out what has been received. If the header was received, its first byte is examined to find out whether this is a control or a data message. If it is a data message, then DI channel is again initiated to receive the data part. For this, first the count of the data bytes is extracted from the header. Also, RFLAG is set to 1. If it is a control message, FREQ is checked to make allocation for the next message. If a buffer is available, the buffer in QNOQ (containing the received message) is shifted to Q3 and a free buffer is shifted to QNOQ, ALLOC is set and the mode bit and SF are reset. The same thing is done when a data message has been completely received (i.e., RFLAG = 1). The flowcharts for the receiver interrupt routines are shown in Figures 4-5 (b) and (c).

- (5) DDR: The functions of this routine are evident from its flowchart alongwith the flowcharts of ACKR and NAKR (Figures 4-6 (a), (b) and (c)).

- (6) USRCV: This routine takes a message from Q4, makes it available to the user for processing and returns the freed buffer to FREQ.

4-7 THE SCHEDULER

There has to be a scheduler which continuously checks various queues and schedules one of the routines USEND, DDT, DDR and USRCV.

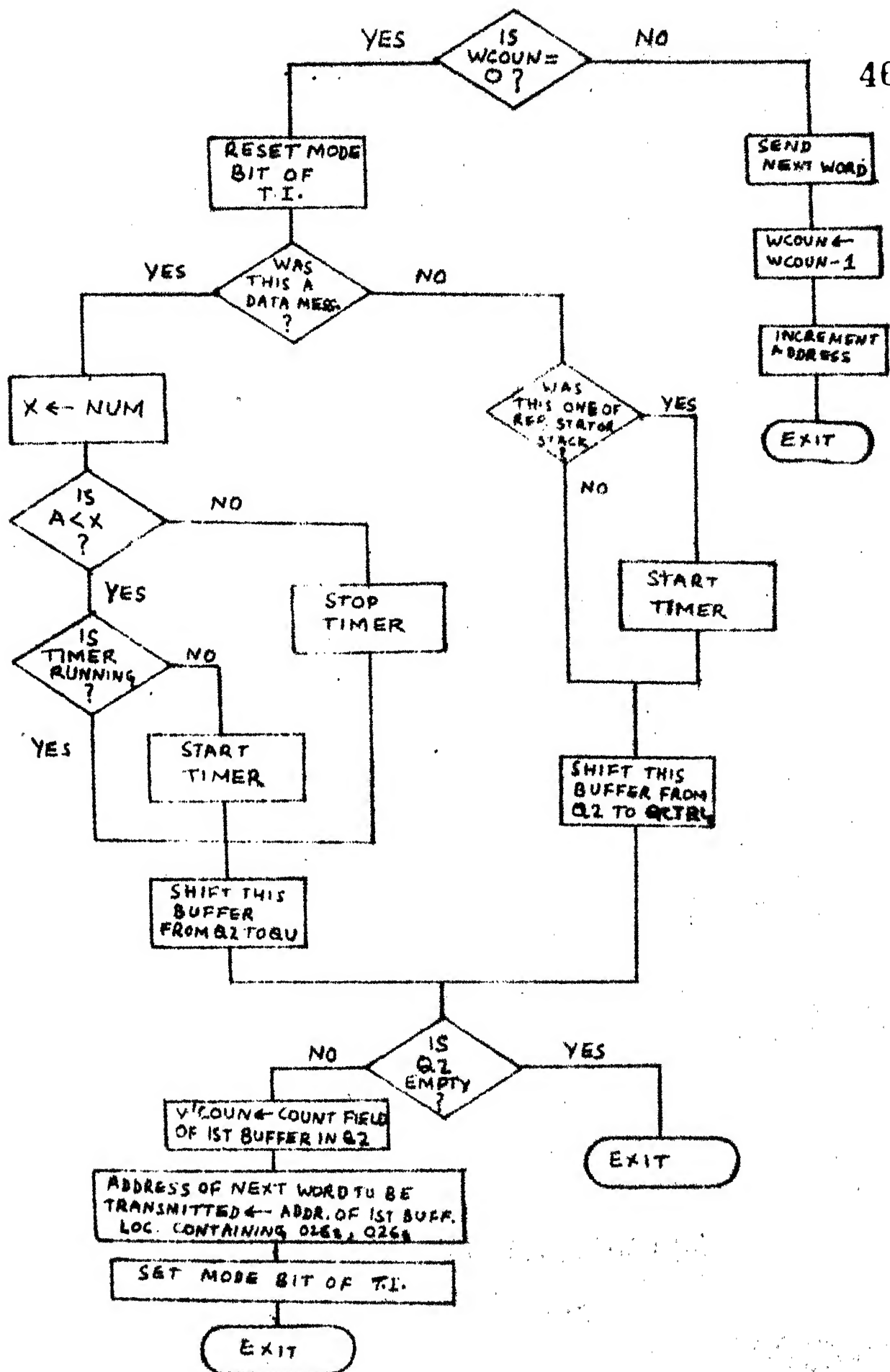


FIG 4.5 : TRANSMITTER INTERRUPT ROUTINE
(a)

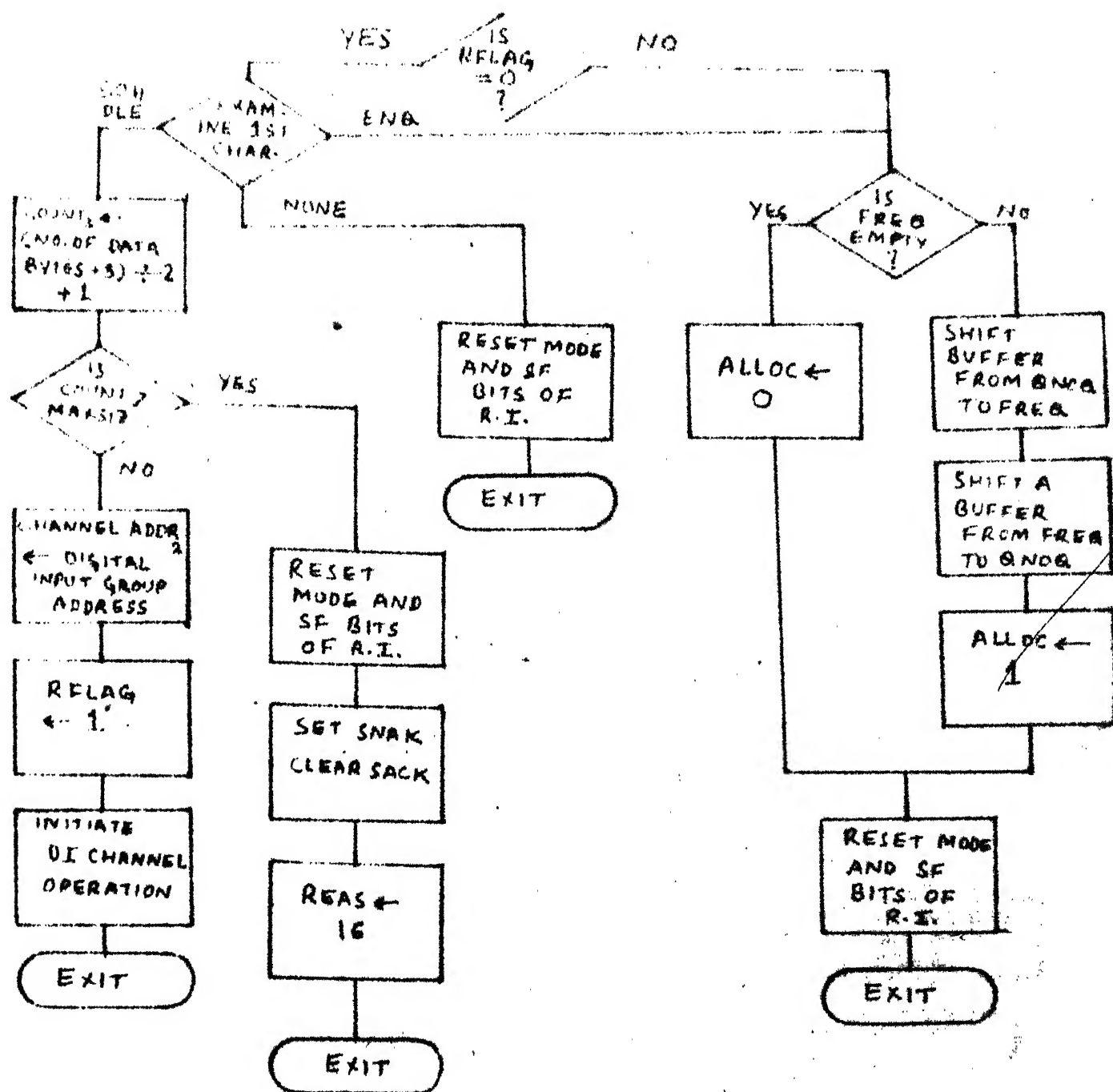


FIG. 4.5 (b): RECEIVER INTERRUPT ROUTINE

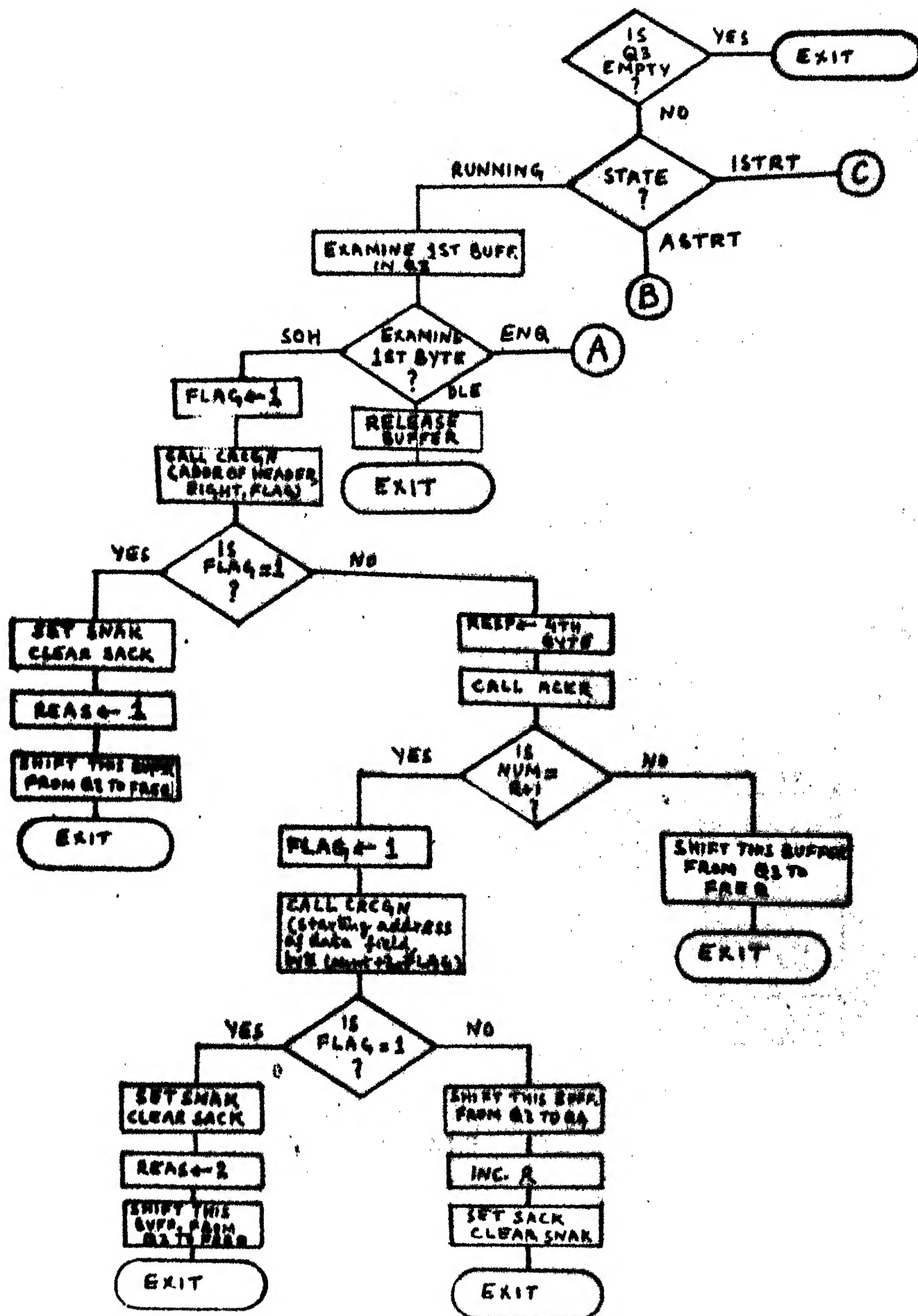


FIG 4-6(A)

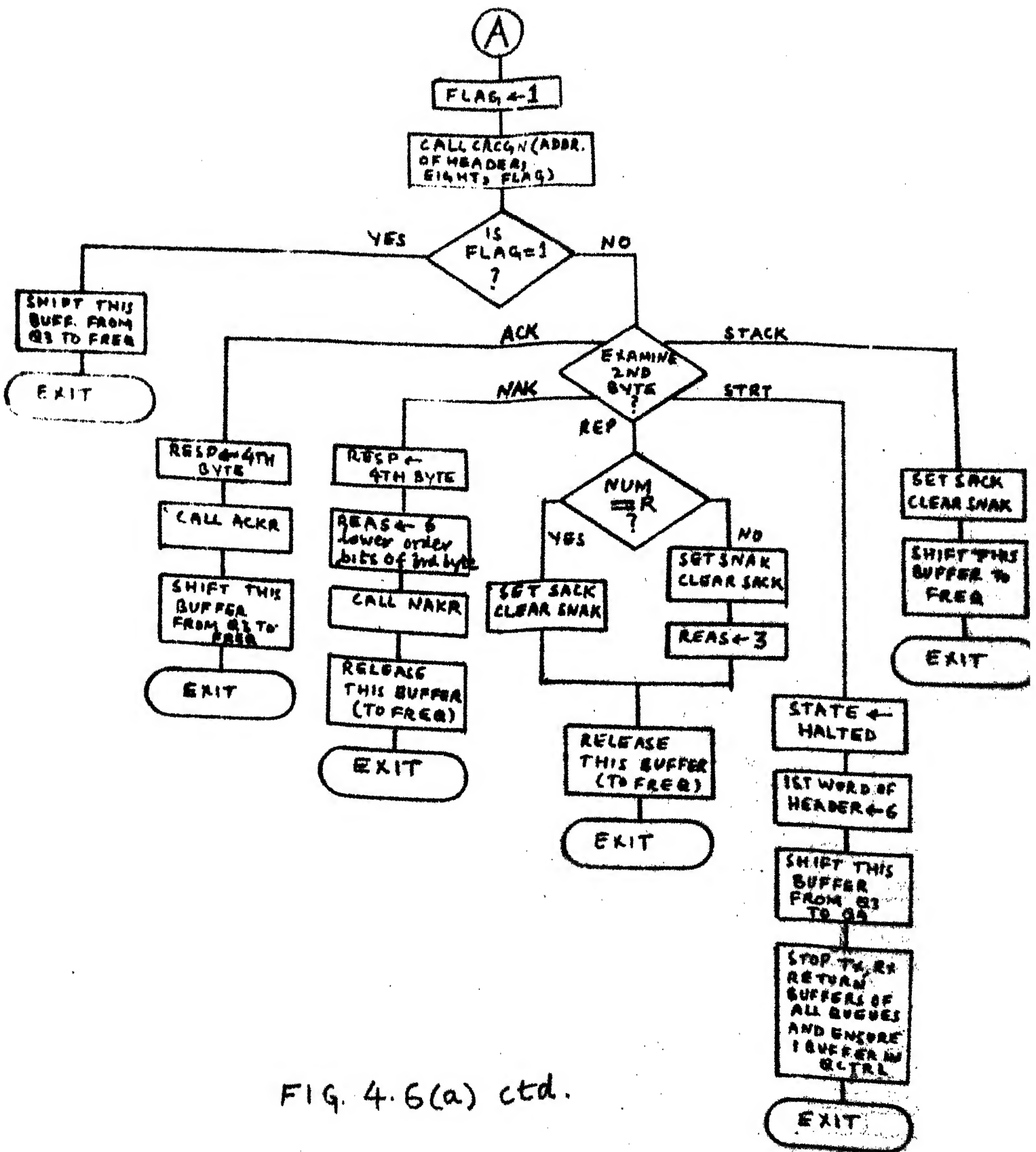


FIG. 4.6(a) ctd.

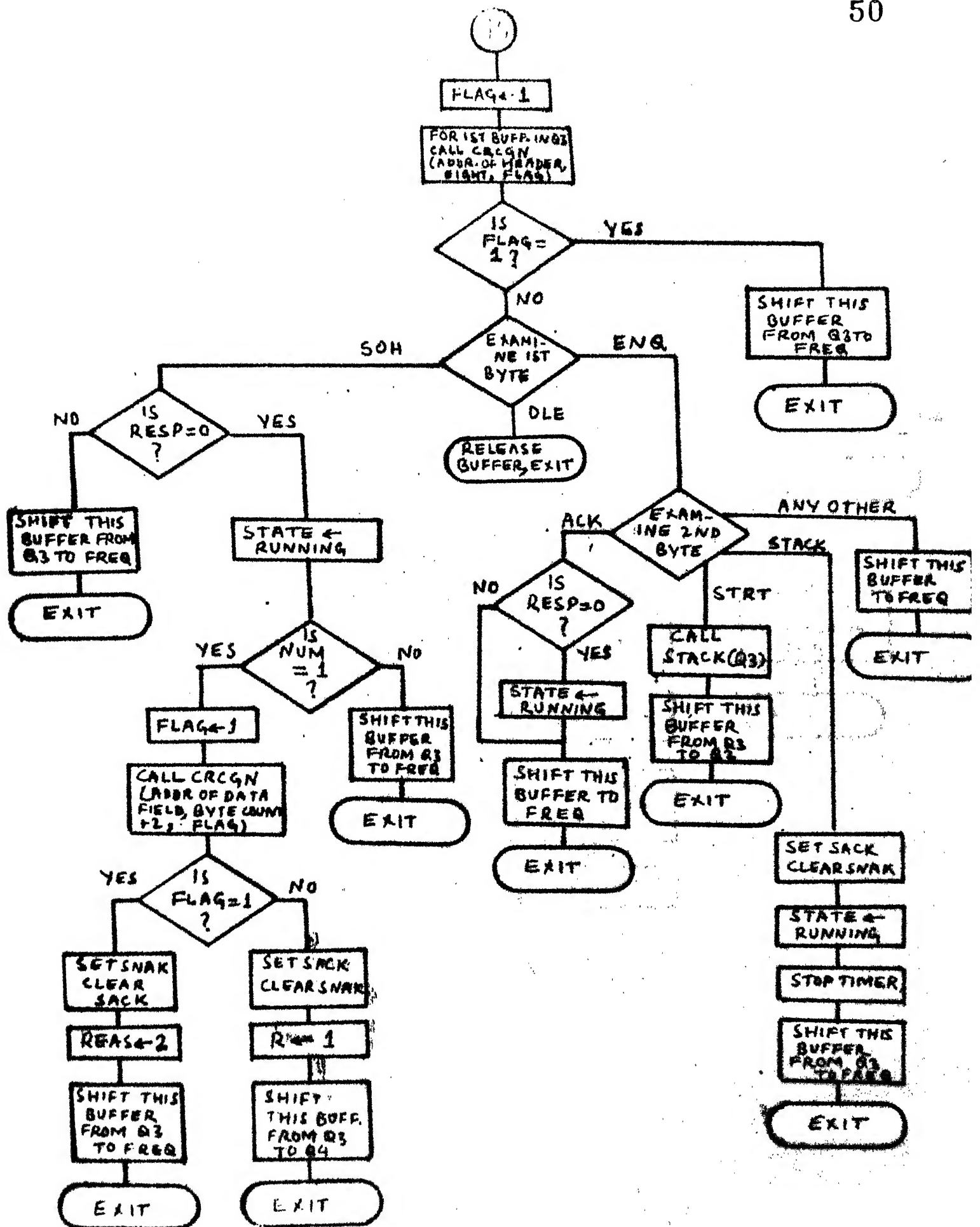


Fig 4.6(a) ctd.

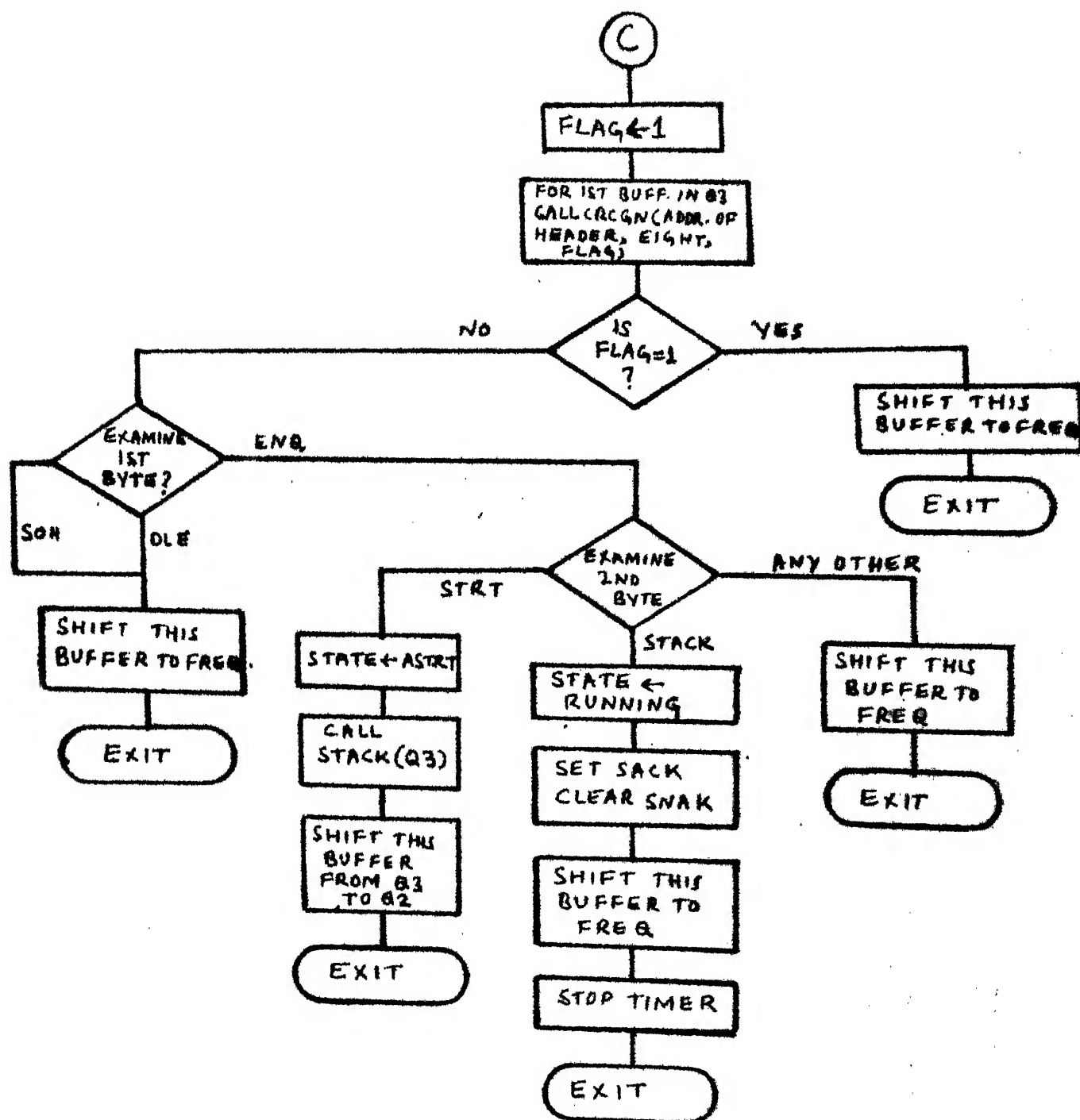


FIG. 4.6 (a). Ctd.

ACKR

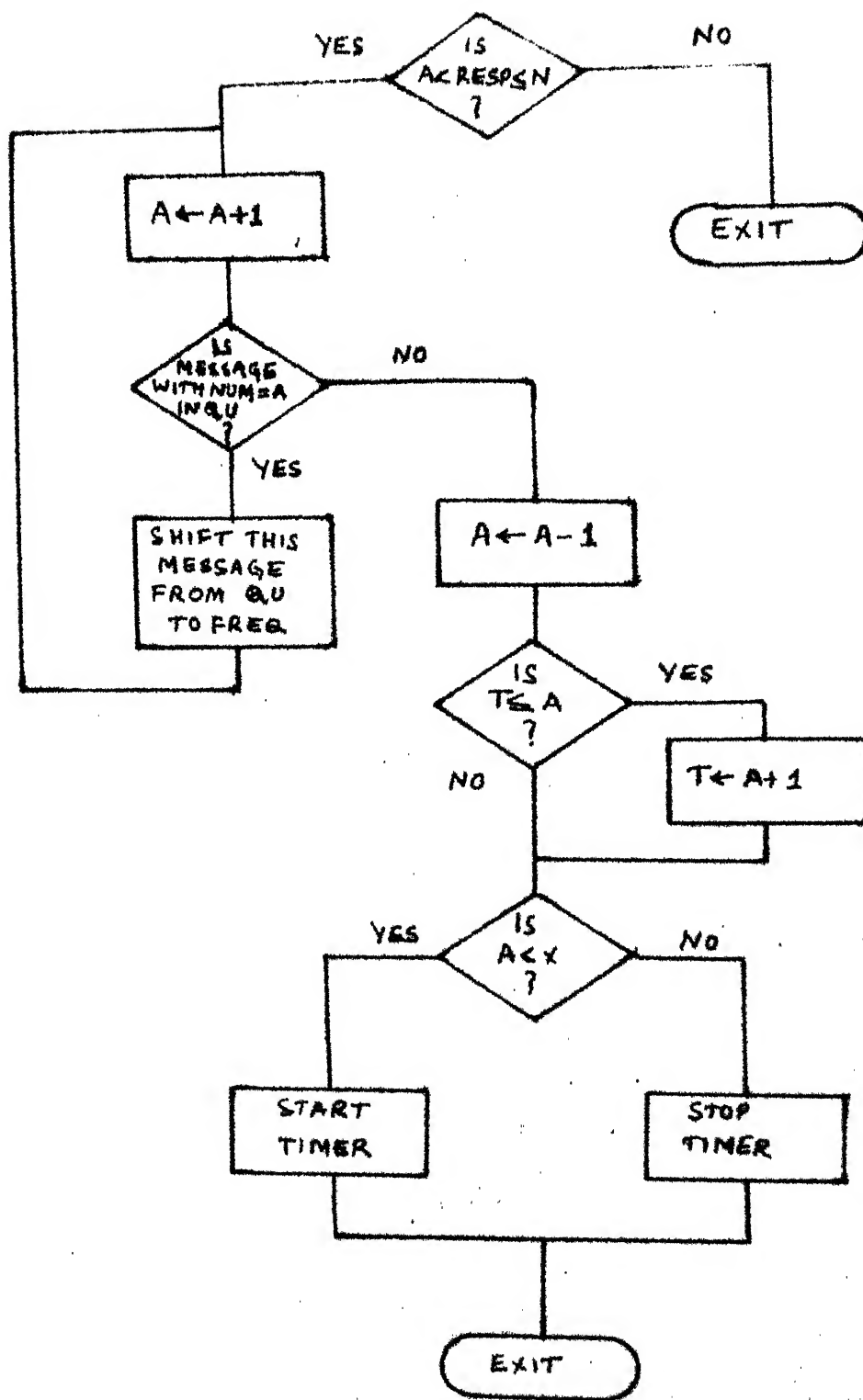


FIG. 4.6(b)

NAKR

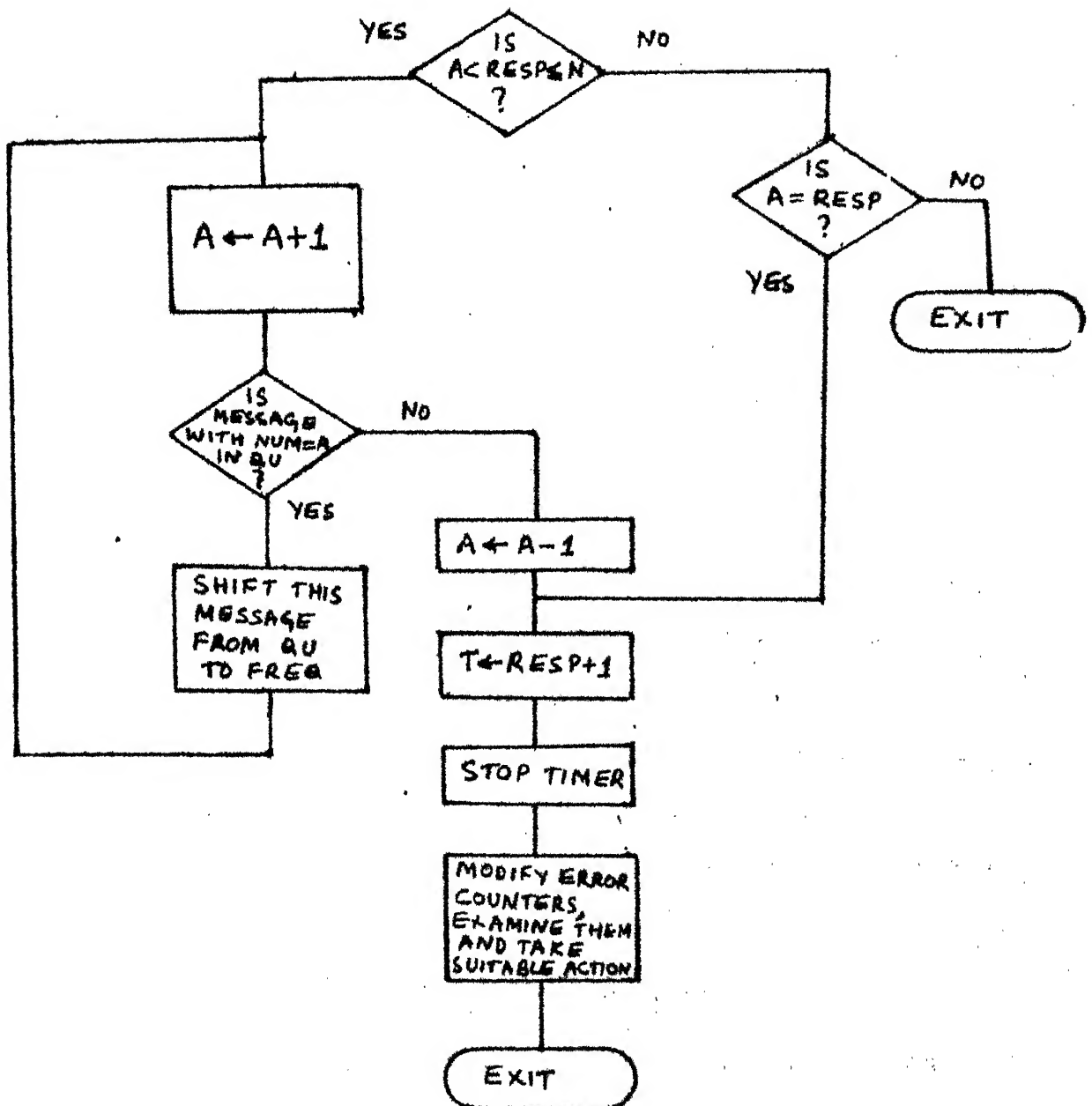


FIG. 4.6 (c)

The scheduling policies have not yet been finalised since this can be done only after a detailed study of the performance statistics of the network. For testing purposes a simple scheduler has been used which schedules the routines in a round-robin fashion. The following checks are made while scheduling:

- (i) USEND is not scheduled if there are more than 5 messages in Q1 or if there are less than 5 messages in FREQ.
- (ii) DDT is scheduled only if there are less than 3 messages in Q2.

The design of the scheduler will change when sufficient statistics about the performance are available.

4-8. TIMER AND ITS INTERRUPT ROUTINE

As soon as a data message or a REP is transmitted completely on the link, a timer is started and after a selected time interval, an interrupt will be generated if no ACK or NAK is received during this period. The interval is selected assigning a value to a variable TIMVAL. The value to be filled in TIMVAL can be calculated as

$$\text{TIMVAL} = 2^{16} - \frac{\text{Interval required in milliseconds}}{.25}$$

The timer interrupt routine examines the STATE of DDMP and takes appropriate actions as shown in the flowchart of Figure 4-7.

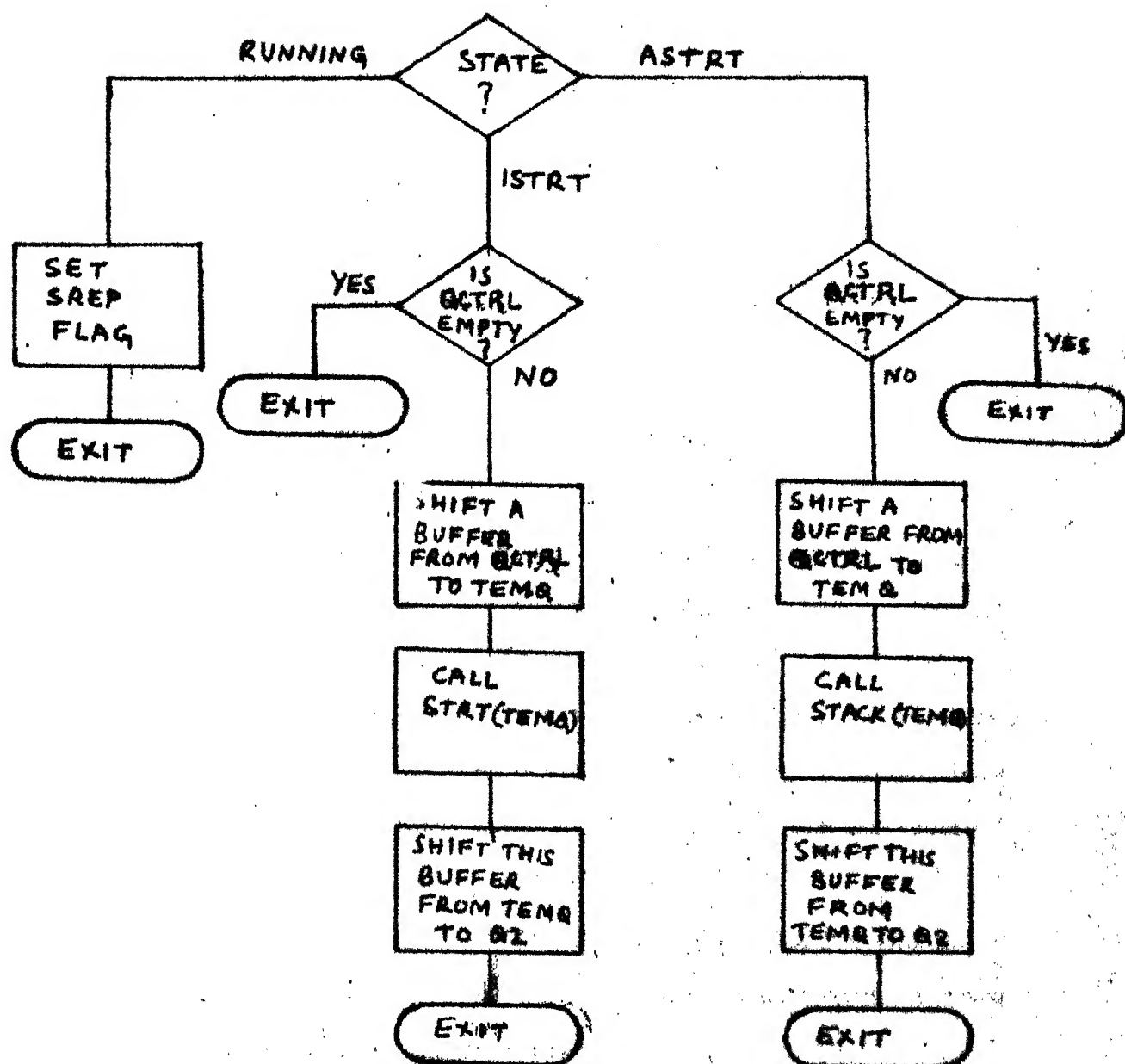


FIG 4.7 : TIMER ROUTINE

CHAPTER 5

CONCLUSION

Our aim during the current phase was to implement the lowest layer protocol on IBM-1800 and MICRO-78 and design the hardware interfaces for the MICRO to DEC link. Since DDCMP was the line protocol used by DEC and its implementation was available on the system, this protocol was chosen for our network and we implemented it on MICRO-78 and IBM-1800. The hardware interfaces designed were found flexible enough to support DDCMP without much problems. However, some other problems were discovered in the design of these interfaces and suitable changes have been made to solve these problems. Moreover, the hardware interfaces of MICRO-78 and TDC-316 though tested gave frequent problems because their fabrication was not very neat and are therefore being refabricated by a professional. The interfaces of IBM-1800 are, however, in a working condition and DDCMP has been tested on them in a loop back mode. On MICRO-78 also, DDCMP has been implemented and tested except for the interrupt routines.

The design of the MICRO to DEC interfaces had to conform to the DQ11 specifications and so it is somewhat different from the other interfaces in the network.

As soon as the hardware interfaces of MICRO-78 and TDC-316 are refabricated, DDCMP can be tested on the IBM-1800-MICRO-78-DEC-10 link and subsequently, higher level protocols can be implemented to provide network facilities between IBM 1800 and

and DEC-10. For TDC-316, the software is already available, and we do not foresee any problems in connecting it to the network.

Since the hardware interfaces on IBM-1800, TDC-316 and MICRO-78 (except the MICRO to DEC side interfaces) were designed before the digital network specifications were available, their design included some features which should now be changed. One change that should be made is to remove the parity generation and checking circuits from these interfaces, since the DQ-11 interface of DEC-10 does not include this feature. Moreover, if any errors do creep in, they are taken care of by the CRC block checks specified in DDUMP. Presently, the CRC generation and checks are performed by software routines, which cause a lot of overhead. To remove this overhead, it is desirable that hardware chips are used to do this work. Another inconsistency between the DDUMP specifications and the earlier hardware interfaces is that the value of SYN character used on the MICRO-78 links to TDC-316 and IBM-1800 is 026_8 as against the value 226_8 specified by DDUMP. This inconsistency can easily be removed by making minor hardware changes.

To increase the data transmission rates, it is suggested that DMA interfaces be used for data transmission and reception. This is particularly essential on the MICRO-78. Another suggestion is that provision be made for varying the baud rate on the links.

APPENDIX I
ERROR MESSAGES

While performing queue operations, two types of fatal errors may be detected and when these errors occur, the system comes to a halt after printing some error messages. DDCMP must be reloaded when these errors occur. The error messages are:

- (1) 'ATTEMPT TO SHIFT FROM AN EMPTY Q. FATAL ERROR
RELOAD SYSTEM.'

This message means that at some stage an attempt was made to take a buffer from some queue which was already empty.

- (2) 'SRCHQ FAILED. FATAL ERROR RELOAD SYSTEM'.

This message means that a buffer was not found in a list when a search was being made for it.

REFERENCES

1. Das, D., "IITK Computer Network - MICRO-78 Hardware Interfaces"
2. DECNET : Digital Data Communications Massage Protocol (DDCMP)
Specification Version 4.0
3. DQ11 Maintenance Manual
4. IBM-1800 Functional Characteristics
5. Krishna, C.V., "IITK Computer Network - IBM-1800 Hardware
Interface"
6. Narayanan, N.S., "IITK Computer Network - TDC-316 Hardware
Interface."